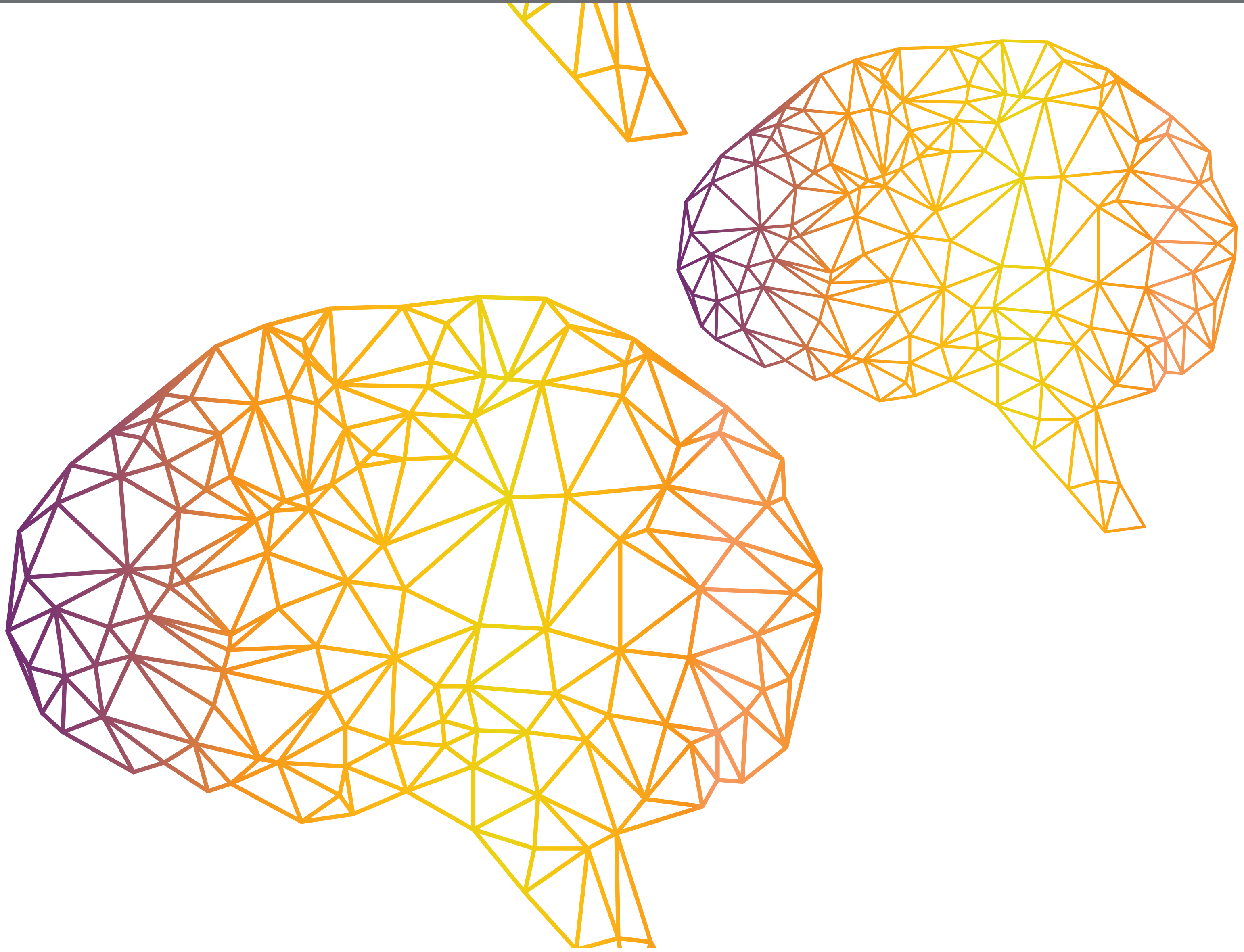
INTRINSICALLY MOTIVATED OPEN-ENDED LEARNING IN AUTONOMOUS ROBOTS

EDITED BY: Vieri Giuliano Santucci, Pierre-Yves Oudeyer, Andrew Barto and Gianluca Baldassarre

PUBLISHED IN: Frontiers in Neurorobotics and Frontiers in Robotics and AI



frontiers Research Topics

ISSN 1664-8714
ISBN 978-2-88963-485-9
DOI 10.3389/978-2-88963-485-9

INTRINSICALLY MOTIVATED OPEN-ENDED LEARNING IN AUTONOMOUS ROBOTS

Topic Editors:
Vieri Giuliano Santucci, Italian National Research Council, Italy
Pierre-Yves Oudeyer, Institut National de Recherche en Informatique et en Automatique (INRIA), France
Andrew Barto, University of Massachusetts Amherst, United States
Gianluca Baldassarre, Italian National Research Council, Italy

Citation: Santucci, V. G., Oudeyer, P.-Y., Barto, A., Baldassarre, G., eds. (2020). Intrinsically Motivated Open-Ended Learning in Autonomous Robots. Lausanne: Frontiers Media SA. doi: 10.3389/978-2-88963-485-9

Table of Contents

05 ***Editorial: Intrinsically Motivated Open-Ended Learning in Autonomous Robots***
Vieri Giuliano Santucci, Pierre-Yves Oudeyer, Andrew Barto and Gianluca Baldassarre

08 ***The Head Turning Modulation System: An Active Multimodal Paradigm for Intrinsically Motivated Exploration of Unknown Environments***
Benjamin Cohen-Lhyver, Sylvain Argentieri and Bruno Gas

31 ***Open-Ended Learning: A Conceptual Framework Based on Representational Redescription***
Stephane Doncieux, David Filliat, Natalia Díaz-Rodríguez, Timothy Hospedales, Richard Duro, Alexandre Coninx, Diederik M. Roijers, Benoît Girard, Nicolas Perrin and Olivier Sigaud

37 ***Cooperative and Competitive Reinforcement and Imitation Learning for a Mixture of Heterogeneous Learning Modules***
Eiji Uchibe

48 ***Stage-Wise Learning of Reaching Using Little Prior Knowledge***
François de La Bourdonnaye, Céline Teulière, Jochen Triesch and Thierry Chateau

61 ***Exploring Criticality as a Generic Adaptive Mechanism***
Miguel Aguilera and Manuel G. Bedia

71 ***Intrinsic Rewards for Maintenance, Approach, Avoidance, and Achievement Goal Types***
Paresh Dhakan, Kathryn Merrick, Iñaki Rañó and Nazmul Siddique

87 ***Evolving Robust Policy Coverage Sets in Multi-Objective Markov Decision Processes Through Intrinsically Motivated Self-Play***
Sherif Abdelfattah, Kathryn Kasmarik and Jiankun Hu

106 ***Robot End Effector Tracking Using Predictive Multisensory Integration***
Lakshitha P. Wijesinghe, Jochen Triesch and Bertram E. Shi

122 ***Learning Inverse Statics Models Efficiently With Symmetry-Based Exploration***
Rania Rayyes, Daniel Kubus and Jochen Steil

138 ***Lifelong Learning of Spatiotemporal Representations With Dual-Memory Recurrent Self-Organization***
German I. Parisi, Jun Tani, Cornelius Weber and Stefan Wermter

153 ***Active Fovea-Based Vision Through Computationally-Effective Model-Based Prediction***
Emmanuel Daucé

168 ***Toward Computational Motivation for Multi-Agent Systems and Swarms***
Md Mohiuddin Khan, Kathryn Kasmarik and Michael Barlow

181 ***Learning a Set of Interrelated Tasks by Using a Succession of Motor Policies for a Socially Guided Intrinsically Motivated Learner***
Nicolas Duminy, Sao Mai Nguyen and Dominique Duhaut

199 ***Boredom-Driven Curious Learning by Homeo-Heterostatic Value Gradients***
Yen Yu, Acer Y. C. Chang and Ryota Kanai

213 ***Ostensive-Cue Sensitive Learning and Exclusive Evaluation of Policies: A Solution for Measuring Contingency of Experiences for Social Developmental Robot***
Hamed Mahzoon, Yuichiro Yoshikawa and Hiroshi Ishiguro

232 ***Artificial Development by Reinforcement Learning Can Benefit From Multiple Motivations***
Günther Palm and Friedhelm Schwenker

238 ***Action Generation Adapted to Low-Level and High-Level Robot-Object Interaction States***
Carlos Maestre, Ghanim Mukhtar, Christophe Gonzales and Stephane Doncieux

259 ***An Embodied Agent Learning Affordances With Intrinsic Motivations and Solving Extrinsic Tasks With Attention and One-Step Planning***
Gianluca Baldassarre, William Lord, Giovanni Granato and Vieri Giuliano Santucci

frontiers in Neurorobotics

EDITORIAL
published: 17 January 2020
doi: 10.3389/fnbot.2019.00115

Editorial: Intrinsically Motivated Open-Ended Learning in Autonomous Robots

Vieri Giuliano Santucci[1], Pierre-Yves Oudeyer[2], Andrew Barto[3] and Gianluca Baldassarre[1]*

[1] Istituto di Scienze e Tecnologie della Cognizione, Consiglio Nazionale delle Ricerche, Rome, Italy, [2] Institut National de Recherche en Informatique et en Automatique (INRIA), Bordeaux, France, [3] College of Information and Computer Sciences, University of Massachusetts, Amherst, MA, United States

Keywords: intrinsic motivation, open-ended learning, robotics, developmental robotics, curiosity driven learning

Editorial on the Research Topic

Intrinsically Motivated Open-Ended Learning in Autonomous Robots

Notwithstanding the important advances in Artificial Intelligence (AI) and robotics, artificial agents still lack the necessary autonomy and versatility to properly interact with realistic environments. This requires agents to face situations that are unknown at design time, to autonomously discover multiple goals/tasks, and to be endowed with learning processes able to solve multiple tasks incrementally and online.

Starting in developmental robotics (Lungarella et al., 2003; Cangelosi and Schlesinger, 2015), and gradually expanding into other fields, intrinsically motivated learning (sometimes called "curiosity-driven learning") has been studied by many researchers as an approach to autonomous lifelong learning in machines (Oudeyer et al., 2007; Schmidhuber, 2010; Barto, 2013; Mirolli and Baldassarre, 2013). Inspired by the ability of humans and other mammals to discover how to produce "interesting" effects in the environment driven by self-generated motivational signals not related to specific tasks or instructions (White, 1959; Berlyne, 1960; Deci and Ryan, 1985), the research in the field of intrinsically motivated open-ended learning aims to develop agents that autonomously generate motivational signals (Merrick, 2010) to acquire repertoires of diverse skills that are likely to become useful later when specific "extrinsic" tasks need to be performed (e.g., Barto et al., 2004; Baldassarre, 2011; Baranes and Oudeyer, 2013; Kulkarni et al., 2016; Santucci et al., 2016).

This Research Topic aims to present state-of-the-art research on open-ended learning in autonomous robots, with a particular focus on systems driven by intrinsic motivations (but not limited to these systems), and augments the information presented at the *Third International Workshop on Intrinsically Motivated Open-ended Learning – IMOL2017*, held in Rome, Italy, 4–6 October 2017. Although the development of autonomous artificial agents is pursued via different kinds of approaches, such as information theory (Klyubin et al., 2008; Martius et al., 2013), epigenetic robotics (Lones et al., 2016), machine learning (Machado et al., 2017), and evolutionary computation (Lehman and Stanley, 2011), intrinsically motivated open-ended learning is today a mature field producing promising research.

The field nevertheless presents many open challenges, the main ones we mention here. A first open issue of central importance for open-ended learning is how an agent should autonomously generate goals and learn policies for achieving them, so that the policies are useful for solving many new tasks that are unknown when the policies are learned. Another open challenge is to design systems that use intrinsic motivations to support learning compact representations of environment states, and hence of goals; in particular, learning compact representations that are relevant for action. Another challenge involves the continuous/discrete representation of goals. It seems plausible that low-level goals (e.g., related to postures that a robot might assume) are encoded in a

OPEN ACCESS

Edited by:
Florian Röhrbein, Technical University of Munich, Germany

Reviewed by:
Lola Cañamero, University of Hertfordshire, United Kingdom

****Correspondence:***
Vieri Giuliano Santucci
vieri.santucci@istc@cnr.it

Received: *25 September 2019*
Accepted: *19 December 2019*
Published: *17 January 2020*

Citation:
Santucci VG, Oudeyer P-Y, Barto A and Baldassarre G (2020) Editorial: Intrinsically Motivated Open-Ended Learning in Autonomous Robots. Front. Neurorobot. 13:115. doi: 10.3389/fnbot.2019.00115

continuous space, whereas high-level goals (e.g., touch, push, hit, reach, grasp) are encoded in a discrete fashion so as to be easily composed to achieve more complex goals. If this is the case, what is the relation between these different types of goal representations? A related problem is how to design architectures that can suitably manage generating goals and learning the related skills, and storing them at different levels of granularity. Then there is the open problem of how to best re-use goals and acquired skills to accomplish novel tasks (exploitation), including how to use previous learning in order to efficiently learn new goals and skills ("transfer learning"), and how to form "chains" of interrelated, hierarchical skills ("curriculum learning"), on the basis of intrinsically motivated processes. Learning different tasks in an open-ended fashion tends to cause catastrophic interference, which is another problem that needs to be addressed. Other important challenges are related to the interaction between intrinsic motivations and other forms of "natural learning" such as social interaction: how this interaction might be connected to the development of higher-level cognitive skills, e.g., language.

The contributions collected for this Topic, reviewed below, not only extend the core research in the field but also tackle some of the open questions mentioned above, with a particular focus on: the autonomous acquisition of skills and motor behaviors; the analysis of architectures and learning signals needed to perform sequences of different tasks; the interaction of artificial agents with their environments through different sensors and actuators; the formation and representation of goals; and the interplay between intrinsic motivations and other learning strategies such as imitation learning. Following is a summary of how the contributions in this collection address these challenges.

In Rayyes et al. the authors propose a system enabling a robot to learn to reach to different points in space by exploiting symmetry properties of the actuators to allow exploration to be limited to only a small part of the configuration space. Maestre et al. tackle skill learning at the level of object manipulation, where low-level and high-level features are extracted through task-agnostic interactions with the environment directed toward learning affordances that, in turn, guide the robot to solve assigned tasks. Baldassarre et al. propose a system that uses intrinsic motivations to learn forward models and affordances, here intended as the probabilities of achieving the goals of the affordance-related actions. In particular, this work examines how active-vision, which allows factoring the environment state into pieces of information related to single objects, can support such learning processes and also facilitate solving extrinsic tasks involving multiple objects through one-step planning. Learning progress might also be used as in the contribution of Uchibe to train multiple skills in parallel. In particular, here transfer learning techniques are combined with "mixture of experts" strategies to develop multiple control modules that are then used to solve control tasks with simulated agents. When learning many different tasks, an agent might try to optimize all of them simultaneously. Abdelfattah et al. leverage intrinsic motivations to develop a method that can cope with multi-objective Markov decision processes, and they compare it to other state-of-the-art algorithms. In real environments, complex tasks might need to be learned through exposure to sequences of simpler tasks. This is a crucial issue for autonomous robotics, which is tackled in the work of Duminy et al. using an active learning approach. These authors propose a new algorithm that allows a robot to autonomously discover how to combine pre-defined primitive motor policies to learn increasingly complex combinations of motor policies. In particular, while it is learning, and agent is able to decide which outcome on which to focus and which exploration strategy to apply, leveraging imitation learning, goal babbling, and strategic learning techniques based on intrinsic motivations.

When an agent can autonomously generate different kinds of goals, task-specific reward functions might be complicated to design since they require significant domain knowledge. Intrinsic motivations might provide general, task-agnostic reward functions able to exploit the inherent properties of different goals. Moreover, as shown in Dhakan et al., these reward functions can be used as building blocks to generate sequences of tasks enabling more complex behaviors to be learned. A well-known problem, mentioned above, related to learning multiple tasks is that of catastrophic forgetting. This problem might be even harder to tackle for artificial agents that have to perform life-long learning in complex and unknown environments. Instead of constraining the set of inputs at design time, in Parisi et al., the authors propose a dual-memory self-organizing architecture with two growing recurrent networks that in parallel learn episodic memory and semantic memory, expanding their structures in response to novel sensory experiences.

Vision plays an important role in many aspects related to autonomous learning and exploration. de La Bourdonnaye et al. followed a developmental perspective and built an artificial system that learns to reach for objects in different locations in the environment by leveraging a weakly-supervised stage-wise procedure. Learning to reach is divided into three tasks: learn to fixate objects, learn hand-eye coordination (learn to fixate on the end-effector), and learn to use the previously acquired knowledge to perform reaching to different locations. Visuo-motor coordination is also tackled by Wijesinghe et al., where predictive models are used to guide a humanoid robot in learning to track its hand and other movements in the visual field without the use of any forward kinematics or pre-defined visual feature descriptors. The use of prediction in multi-sensory integration allows a better incorporation of proprioceptive and visual cues and leads to the development of emergent properties similar to those of human hand-eye coordination. Task-agnostic motivations such as information gain are used in the work of Dauce to drive action selection and the exploration of visual inputs: promising results are shown, highlighting how compression strategies might improve both performance in visual recognition and efficiency of the system thanks to reduced computational costs. Autonomous exploration is the focus of Cohen-Lhyver et al. These authors underline the important role of attention, which they claim can be considered as a sort of intrinsic motivation. They implemented this notion in a humanoid robot and showed how two components (congruence and reduction of uncertainty) can be used to explore new environments following audio-visual inputs encoded at a semantic level.

Similar to appealing to information gain, exploration drive by "criticality" can be used to generate autonomous behaviors. Aguilera and Bedia explore this connection through conceptual models that exploit maximum entropy to drive agents toward critical points (e.g., transition points between different kinds of behaviors). Finally, Mahzoon et al. focus on the development of social robots. Within a developmental perspective, these authors address problems related to training real-world robots by presenting two new algorithms that improve a robot's performance in terms of learning efficiency, complexity of the learned behaviors, and predictability of the robot's behavior.

In addition to the aforementioned 14 original research articles, four additional papers have been published within this Research Topic. The first is a methodological article in which Yu et al. present an algorithm that takes into account the constructive interplay between boredom and curiosity, giving rise to effective exploration and forward model learning. The second is a review article in which Khan et al. examine the motivational systems used in computational models to build agents capable of autonomous goal generation and task learning. The authors then investigate how these strategies might be transferred to multi-agent systems and swarms, highlighting the current state-of-the-art and future key challenges. The last two papers are perspective articles. In the first of these, Doncieux et al. argue that a key issue for an agent performing open-ended learning is not only the problem of maximizing the rewards related to the different tasks, but also the problem of building proper representations of the states and the actions describing the tasks themselves. The authors present a conceptual framework to address this crucial issue, underlining the central role of intrinsic motivations. In the second article, Palm and Schwenker analyse the use of reinforcement learning (RL) in the field of developmental robotics, describing its strengths and weaknesses with respect to some specific problems that arise in the field. The authors suggest that multi-objective RL might face some of the problems they listed and that leveraging multiple motivations can improve RL agents' learning performance.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

FUNDING

The research and editorial activity related to this topic, has been partially funded from the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement no. 713010 (GOAL-Robots—Goal-based Open-ended Autonomous Learning Robots).

REFERENCES

Baldassarre, G. (2011). "What are intrinsic motivations? a biological perspective," in *Proceedings of the International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob-2011)* (New York, NY; Frankfurt am Main: IEEE), E1–E8.

Baranes, A., and Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robot. Auton. Syst.* 61, 49–73. doi: 10.1016/j.robot.2012.05.008

Barto, A. G. (2013). "Intrinsic motivation and reinforcement learning," in *Intrinsically Motivated Learning in Natural and Artificial Systems* (Berlin; Heidelberg: Springer), 17–47.

Barto, A. G., Singh, S., and Chentanez, N. (2004). "Intrinsically motivated learning of hierarchical collections of skills," in *Proceedings of the 3rd International Conference on Development and Learning* (La Jolla, CA), 112–119.

Berlyne, D. E. (1960). *Conflict, Arousal, and Curiosity*. New York, NY: McGraw Hill.

Cangelosi, A., and Schlesinger, M. (2015). *Developmental Robotics: From Babies to Robots*. Cambridge, MA: MIT Press.

Deci, E. L., and Ryan, R. M. (1985). *Intrinsic Motivation and Self-Determination in Human Behavior*. New York, NY: Plenum Press.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2008). Keep your options open: an information-based driving principle for sensorimotor systems. *PLoS ONE* 3:e4018. doi: 10.1371/journal.pone.0004018

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. (2016). "Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation," in *Advances in Neural Information Processing Systems* (Bacelona), 3675–3683.

Lehman, J., and Stanley, K. O. (2011). Abandoning objectives: evolution through the search for novelty alone. *Evol. Comput.* 19, 189–223. doi: 10.1162/EVCO_a_00025

Lones, J., Lewis, M., and Cañamero, L. (2016). From sensorimotor experiences to cognitive development: investigating the influence of experiential diversity on the development of an epigenetic robot. *Front. Robot. AI* 3:44. doi: 10.3389/frobt.2016.00044

Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *Connect. Sci.* 15, 151–190. doi: 10.1080/09540090310001655110

Machado, M. C., Bellemare, M. G., and Bowling, M. (2017). "A laplacian framework for option discovery in reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2295–2304.

Martius, G., Der, R., and Ay, N. (2013). Information driven self-organization of complex robotic behaviors. *PLoS ONE* 8:e63400. doi: 10.1371/journal.pone.0063400

Merrick, K. E. (2010). A comparative study of value systems for self-motivated exploration and learning by robots. *IEEE Trans. Auton. Ment. Dev.* 2, 119–131. doi: 10.1109/TAMD.2010.2051435

Mirolli, M., and Baldassarre, G. (2013). "Functions and mechanisms of intrinsic motivations," in *Intrinsically Motivated Learning in Natural and Artificial Systems*, eds G. Baldassarre and M. Mirolli (Berlin; Heidelberg: Springer), 49–72.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evol. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2016). Grail: a goal-discovering robotic architecture for intrinsically-motivated learning. *IEEE Trans. Cogn. Dev. Syst.* 8, 214–231. doi: 10.1109/TCDS.2016.2538961

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Trans. Auton. Ment. Dev.* 2, 230–247. doi: 10.1109/TAMD.2010.2056368

White, R. W. (1959). Motivation reconsidered: the concept of competence. *Psychol. Rev.* 66:297. doi: 10.1037/h0040934

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH
published: 21 September 2018
doi: 10.3389/fnbot.2018.00060

The Head Turning Modulation System: An Active Multimodal Paradigm for Intrinsically Motivated Exploration of Unknown Environments

Benjamin Cohen-Lhyver, Sylvain Argentieri* and Bruno Gas

CNRS, Institut des Systèmes Intelligents et de Robotique, Sorbonne Université, Paris, France

Over the last 20 years, a significant part of the research in exploratory robotics partially switches from looking for the most efficient way of exploring an unknown environment to finding what could motivate a robot to autonomously explore it. Moreover, a growing literature focuses not only on the topological description of a space (dimensions, obstacles, usable paths, etc.) but rather on more semantic components, such as multimodal objects present in it. In the search of designing robots that behave autonomously by embedding life-long learning abilities, the inclusion of mechanisms of attention is of importance. Indeed, be it endogenous or exogenous, attention constitutes a form of intrinsic motivation for it can trigger motor command toward specific stimuli, thus leading to an exploration of the space. The Head Turning Modulation model presented in this paper is composed of two modules providing a robot with two different forms of intrinsic motivations leading to triggering head movements toward audiovisual sources appearing in unknown environments. First, the Dynamic Weighting module implements a motivation by the concept of Congruence, a concept defined as an adaptive form of semantic saliency specific for each explored environment. Then, the Multimodal Fusion and Inference module implements a motivation by the reduction of Uncertainty through a self-supervised online learning algorithm that can autonomously determine local consistencies. One of the novelty of the proposed model is to solely rely on semantic inputs (namely audio and visual labels the sources belong to), in opposition to the traditional analysis of the low-level characteristics of the perceived data. Another contribution is found in the way the exploration is exploited to actively learn the relationship between the visual and auditory modalities. Importantly, the robot—endowed with binocular vision, binaural audition and a rotating head—does not have access to prior information about the different environments it will explore. Consequently, it will have to learn in real-time what audiovisual objects are of "importance" in order to rotate its head toward them. Results presented in this paper have been obtained in simulated environments as well as with a real robot in realistic experimental conditions.

Keywords: multimodal perception, attention, motivation, active learning, binaural audition

OPEN ACCESS

Edited by:
*Pierre-Yves Oudeyer,
Institut National de Recherche en Informatique et en Automatique (INRIA), France*

Reviewed by:
*Pablo Lanillos,
Technische Universität München, Germany*
*Nicolas Cuperlier,
Université de Cergy-Pontoise, France*

****Correspondence:***
*Sylvain Argentieri
sylvain.argentieri@sorbonne-universite.fr*

Received: *31 May 2018*
Accepted: *30 August 2018*
Published: *21 September 2018*

Citation:
Cohen-Lhyver B, Argentieri S and Gas B (2018) The Head Turning Modulation System: An Active Multimodal Paradigm for Intrinsically Motivated Exploration of Unknown Environments. Front. Neurorobot. 12:60. doi: 10.3389/fnbot.2018.00060

1. INTRODUCTION

One of the most critical and important task humans are able to do is to explore unknown environments, topologically or semantically, while being able to create internal representations of them for localization in it and interaction with it. Such cerebral representations, or maps as it is often referred to O'Keefe and Nadel (1978) and Cuperlier et al. (2007), enable humans and animals in general to gather and organize perceptual cues (visual, acoustic, tactile, olfactory, proprioceptive…) in semantic components. In parallel, in the mobile robotics community, exploration of unknown environments has been one of the most important fields studied, back to the artificial turtles of Walter (1951) and later to the vehicles of Braitenberg (1986). Indeed, being able for a mobile robot to simultaneously (i) map the world it is exploring, (ii) locate itself in it, and (iii) trigger relevant motor actions for further exploration (i.e., the three key tasks to perform in an exploration scheme according to Makarenko et al., 2002), has shown to be a hard, but critical for robots' existence, problem to solve. While many artificial systems have been implemented with the sole purpose of *exploring the most of an environment* with only efficiency as a goal (Smith et al., 1987; Henneberger et al., 1991; Montemerlo et al., 2002; Carrillo et al., 2015), some more recent algorithms emerged on the basis of the precursor works of Berlyne (1950, 1965), who stated that *Motivation* is a fundamental mechanism in spontaneous exploratory behaviors in humans. Following this principle, exploration would not be driven by a goal defined by an external agent (such as the human experimenter) but rather by internal goals defined by the robot itself, that is *intrinsic* motivations (Ryan and Deci, 2000; Oudeyer and Kaplan, 2008). Amongst them are the motivations by *Curiosity*, first mathematically modeled by Schmidhuber (1991), by *Uncertainty* (Huang and Weng, 2002), by *Information gain* (Roy et al., 2001), or by *Empowerment* (Capdepuy et al., 2007). Intrinsic motivation has extensively been used during the last 20 years in several powerful systems, in particular by Oudeyer et al. (2007) with the development of the Independent Adaptive Curiosity algorithm (IAC) and the later updated systems (R-IAC, Baranes and Oudeyer, 2009 and SAGG-RIAC, Baranes and Oudeyer, 2010). Systems based on such motivations to explore/understand an environment incorporate in particular the notion of *reward*, a principle that is of high importance in learning in primates and humans (Rushworth et al., 2011). As such, these systems are particularly suited for adaptive life-long learning robots for they bring to them wider motivations to react to their environments: instead of compelling the robot to "*explore as quickly as possible every inch of the room*", it becomes closer to "*just be curious*". But beyond the topological characteristics of unknown environments, their content also provides valuable information for the robots internal representation of the world (object formation, their affordance, etc.). Then, while one of the most predominant issue in driving topological exploration is to decide what is the next point or area to explore, semantical exploration can be also introduced to determine what is the next component to discover. Such considerations are close to attentional behaviors, which have also been extensively studied (Downar et al., 2000; Hopfinger et al., 2000; Corbetta and Shulman, 2002; Corbetta et al., 2008; Petersen and Posner, 2012).

Among others, saliency is known to be a key feature in attention thanks to its sensitivity to discontinuity in perceived data. A significant literature can be found on saliency-driven exploration: eye saccades modelization (Itti et al., 1998; Oliva et al., 2003; Le Meur and Liu, 2015), detection of auditory salient events (Kayser et al., 2005; Duangudom and Anderson, 2007), or audiovisual objects exploration (Ruesch et al., 2008; Tsiami et al., 2016). However, most of these models propose either a solely off-line solution requiring prior training from large databases, or an immutable saliency characterization of events. Moreover, the fact that these models only deal with the low-level characteristics of the perceived data leads often to an absence of wider context inclusion, be it through a form of memory, or through the semantics of the events. In addition, saliency can somehow differ from importance, depending on the task to accomplish: attention can be driven by behaviorally important but not salient stimuli while, on the other hand, very salient stimuli but showing no behavioral importance can be disregarded by the attentional networks (Corbetta and Shulman, 2002; Indovina and Macaluso, 2007). However, it is worth mentioning the interesting feature of the multimodal model of salience of (Ruesch et al., 2008) as the implementation of an additional inhibition map to the ones already used for saliency. Such map promotes the exploration of unknown parts of the environments and avoids deadlock situations caused by local minima. This has also to be brought close to the notion of motivations for exploration mentioned above since a form of Curiosity is here implemented.

In this paper is presented a computational system, The Head Turning Modulation system (HTM), which aims at giving a mobile robot endowed with binaural hearing, binocular vision and a rotating head, the ability to decide which audiovisual sources present in unknown environments are worth the robot's attention. The principle of attention mentioned in this paper is based on the prime definition originating from James (1890): "Everyone knows what attention is. It is the taking possession by the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought". More particularly, the proposed HTM system is dedicated to the implementation of an overt and endogenous (Driver and Spence, 1998; Le Meur et al., 2006) attentional reaction: *the head turning*. This reaction, known to be one of the attentional behavior involved in the mechanism of *attention reorientation* to unpredictable stimuli (Thompson and Masterton, 1978; Corbetta et al., 2008; Corneil et al., 2008), aims at bringing the visual sensors in front of the sources of interest hence enabling the robot to gather and analyze additional data. In addition, the HTM system provides the robot with an adaptive enough online learning behavior so that it can endlessly integrates new useful information to its self-created audiovisual database. However, this learning relying intensively upon the triggering of head movements, it is also necessary for the robot to understand when this knowledge is robust and relevant enough, thus not requiring further motor reaction. The HTM is part of a much wider system, implemented as the TWO!EARS software[1], which aims at providing a computational framework for modeling active exploratory listening that assigns meaning to auditory scenes. More precisely, it consists in perceiving and analyzing a multimodal world through a paradigm that combines a classical bottom-up signal-driven processing step together with a top-down cognitive feedback. In there, the HTM is in charge of building an internal semantic map of the explored environment, made of localized audiovisual objects coupled with their respective semantic importance, the so-called congruence.

In comparison with other works, the proposed system is described as a *real-time* (Huang et al., 2006) and *online* behavioral unit, which is always able to learn new situations while also taking advantage of its previous experience of the past environments. In terms of architecture, the proposed system receives data from several "experts" from the TWO!EARS software, i.e., computational elements specialized in very particular tasks, such as the identification of sounds or images. It means that the HTM system is placed right after these experts, and thus receives already highly interpreted data. Two main parts constitute the system: an attentional component, the *Dynamic Weighting* module (Walther and Cohen-Lhyver, 2014), and a learning component, the *Multimodal Fusion & Inference* module (Cohen-Lhyver et al., 2015). On the one hand, the DW module is dedicated to the analysis of perceived audiovisual objects through the concept of *Congruence*, defined as a semantic saliency and rooted in the principle of optimal incongruity (Hunt, 1965). The DW module implements a form of motivation by surprise for it favors unexpected audiovisual events. On the other hand, the MFI module learns the association between auditory and visual data in order to make the notion of *multimodal object* arise from potentially erroneous data of the aforementioned experts. The MFI module implements a form of motivation by reduction of uncertainty for it aims at consolidating as much as needed its knowledge about the audiovisual objects that the robot encounters. This learning serves two purposes. First, it might improve the robustness and reliability of the classification (Droniou et al., 2015). Secondly, it allows the system to perform missing information inference (Bauer and Wermter, 2013), as when an object is placed behind the robot thus having only access to the auditory information.

The paper is organized as follows. To begin with, the overall TWO!EARS framework, together with the notations used all along the paper, are introduced in a first section. On this basis, the overall HTM system is thoroughly presented in a second section: after a short insight into the HTM system architecture, the way the DW module and the MFI module operate is formalized. This section also presents their respective evaluation in simulated conditions. Then, the combination of the two modules is investigated and the evaluation of the approach in real experimental conditions, that is including a real robot in a real environment, is made. Finally, a conclusion ends the paper.

[1]http://www.twoears.eu

2. CONTEXT AND NOTATIONS

This section presents the context in which is rooted the proposed HTM system. All the forthcoming development has taken place inside a specific computational architecture aiming at modeling an integral, multimodal, intelligent and active auditory perception and experience. This model physically uses two human-like ears and visual inputs to make a mobile robot able to interactively explore unknown environments, see Two!Ears (2016b). Among other applications, this modular architecture targets evaluation of bottom-up audiovisual processing coupled with top-down cognitive processes. The proposed HTM system relies also on this top-down and bottom-up paradigm providing the robot with a reliable internal representation of its audiovisual environment. To begin with, a short overview of the overall architecture is proposed in a first subsection. A second subsection introduces the notations used all along the paper, together with all the notions required to understand the HTM system.

2.1. Global Framework

All the forthcoming developments have been conducted inside the multilayer TWO!EARS architecture, see **Figure 1A**. This figure highlights two different pathways: first, a classical bottom-up processing way, where raw data coming from the sensors (microphones and cameras) are first analyzed (features extraction step), processed (through some specialized pattern recognition algorithms) and interpreted (representation and decisional layers). All of the above is computed by dedicated Knowledge Sources (KS). The main contribution of this architecture is that all these layers are highly and dynamically parameterizable: for instance, most of the feature extractions parameters (for audio data, one could cite the number of Gammatone filters used, their repartition on the frequency scale, etc.) can be changed on the fly. In general, the decision to change parameters comes from upper layers, resulting in a top-down pathway, also involving decisions concerning the movement of the robot itself. Such decisions concerning the robot actions are of particular importance, especially when dealing with attention reorientation and scene understanding for they add adaptability to new and unpredictable events.

The HTM system inside the TWO!EARS architecture shown in **Figure 1B** is implemented as a Knowledge Source (KS). It gets data from other KSs available in the architecture through a blackboard (Schymura et al., 2014) (which can be seen, with a rough simplification, as a data structure), and provides as an output a proposition for a motor command, together with an interpreted representation of the robot's world. One originality of the approach is that the HTM system is placed behind other KS, thus not working directly with the features extracted from the raw audio and visual signals. All of the KSs the HTM relies upon contribute to the scene analysis and are fused by the HTM into a representation of the world that spans wider in time than the one provided by the individual KSs. This representation is made of all the unknown environments explored by the robot, each of them being characterized by the audiovisual objects observed there in an allocentric representation, coupled with an additional semantic layer formalized through the notion of Congruence.

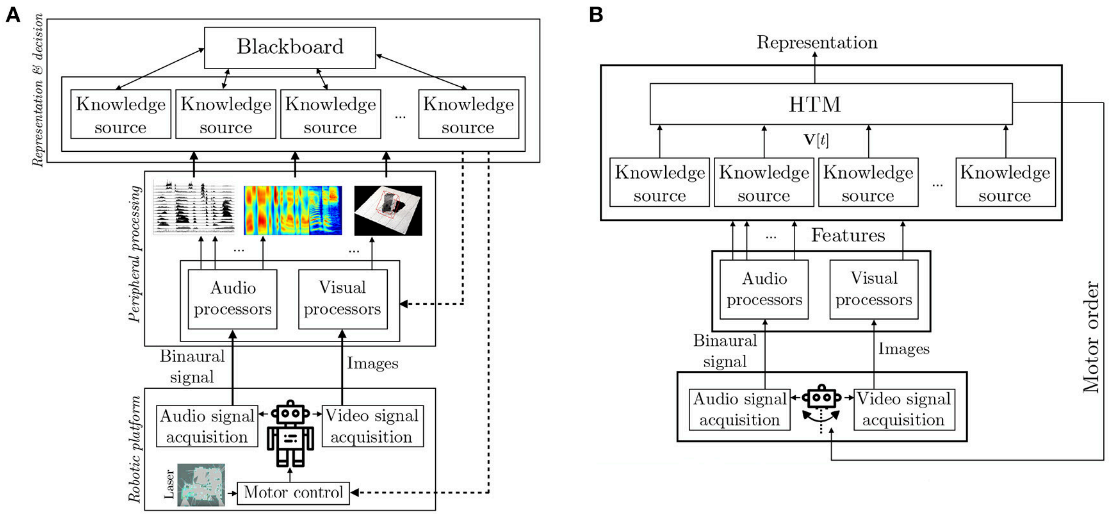


FIGURE 1 | TWO!EARS architecture. **(A)** On the basis on audio and visual data, features are extracted to provide a compact description of the data. Several audio and visual experts (or Knowledge Sources, KS) exploit these features to analyze the signals. Each KS is specialized in one task: recognition of one type of sound, localization, separation, etc. All experts share their knowledge through a blackboard system, thus producing an internal representation of the world. On this basis, the overall system (but also individual KSs) can decide to modulate either the feature extraction step, or the action of the robot. The proposed HTM system, implemented as a KS, is–among others–responsible for this last modulation. **(B)** Focus on the implementation of the HTM system inside this architecture.

The data used by the HTM, together with their notations are described in the following section.

2.2. Definitions and Notations

The HTM system only relies upon KSs outputs to analyze the unknown environments the robot explores. These KSs are classification experts specialized in the recognition of audio or visual frames (Two!Ears, 2016a), classified in terms of audio classes $\mathtt{c}_i^a$, with $i = 1, \dots, N_a$ (such as $\mathtt{c}_i^a \in \{\mathtt{voice}, \mathtt{barking}, \mathtt{yelling}, \dots\}$) or visual classes $\mathtt{c}_k^v$, with $k = 1, \dots, N_v$ (such as $\mathtt{c}_k^v \in \{\text{DOG}, \text{BABY}, \text{MALE PERSON}, \dots\}$) with N_a and N_v the number of audio and visual classes, respectively. All classifiers are mutually independent, each providing a probability $p_i^a[t]$ and $p_k^v[t]$ for the frame at time t to belong to the class they represent. All these probabilities are regrouped by modality in the two vectors $\mathbf{P}^a[t] = (p_1^a[t], \dots, p_{N_a}^a[t])$ and $\mathbf{P}^v[t] = (p_1^v[t], \dots, p_{N_v}^v[t])$. In addition, the TWO!EARS architecture provides N_θ localization experts (May et al., 2011; Ma et al., 2015), aiming at localizing audio and/or visual events in the horizontal plane with respect to the robot. Each of them outputs a probability $p_{\theta_u}^a[t]$ and $p_{\theta_u}^v[t]$, with $u = 1, \dots, N_\theta$, for an audio and/or visual event to originate from the azimuth θ_u^a or θ_u^v (by convention, $\theta = 0°$ corresponds to an event placed in front of the robot). All these probabilities are gathered into the audio and visual localization vectors $\mathbf{\Theta}^a[t] = (p_{\theta_1}^a[t], \dots, p_{\theta_{N_\theta}}^a[t])$ and $\mathbf{\Theta}^v[t] = (p_{\theta_1}^v[t], \dots, p_{\theta_{N_\theta}}^v[t])$. In practice, all these classifiers outputs are regrouped into a single vector $\mathbf{V}[t]$ constituting the sole HTM system input, with

$$\begin{aligned} &\mathbf{V}[t] = (\mathbf{P}[t], \mathbf{\Theta}[t]), \\ &\text{with } \mathbf{P}[t] = (\mathbf{P}^a[t], \mathbf{P}^v[t]) \text{ and } \mathbf{\Theta}[t] = (\mathbf{\Theta}^a[t], \mathbf{\Theta}^v[t]). \end{aligned} \tag{1}$$

From $\mathbf{V}[t]$, the HTM model attempts to build a stable and reliable internal representation of the world, environment by environment. Such a representation is obtained by transforming an audio and/or visual event Ψ_j objectively present in the environment at azimuth $\theta(\Psi_j)$ and belonging to the ground truth audiovisual class $\mathtt{c}(\Psi_j) = \{\mathtt{c}^a(\Psi_j), \mathtt{c}^v(\Psi_j)\}$, into an object o_j perceived by the robot, i.e.,

$$\begin{aligned} &\Psi_j = \{\theta(\Psi_j), \mathtt{c}(\Psi_j)\} \longrightarrow o_j = \{\widehat{\theta}(o_j), \widehat{\mathtt{c}}(o_j)\}, \\ &\text{with } \widehat{\theta}(o_j) = \begin{cases} \theta_u^a, \text{ with } u = \arg\max_i(p_{\theta_i}^a), \text{ if } \theta_u^a \geq |\theta_{\text{HTM}} - \theta_{\text{FOV}}| \\ \theta_u^v, \text{ with } u = \arg\max_k(p_{\theta_k}^v) \text{ otherwise} \end{cases}, \\ &\text{and } \widehat{\mathtt{c}}(o_j) = \{\widehat{\mathtt{c}}^a(o_j), \widehat{\mathtt{c}}^v(o_j)\}, \end{aligned} \tag{2}$$

where θ_{HTM} and θ_{FOV} represent the current azimuthal head position and the field of view of the camera, respectively. Then, an object o_j is defined by its estimated angular position $\widehat{\theta}(o_j)$ and its estimated audiovisual class $\widehat{\mathtt{c}}(o_j)$ made of the estimated audio class $\widehat{\mathtt{c}}^a(o_j)$ and estimated visual class $\widehat{\mathtt{c}}^v(o_j)$. Equation (2) also indicates that the estimated angular position is obtained from the audio localization experts when the objects are out of the robot sight; otherwise, visual localization experts are exploited. Because of localization and/or classification errors, the object o_j might differ from the corresponding Ψ_j. As an example, **Figure 2** plots as a function of temporal frames experimental data from three audio classifiers outputs corresponding to the audio classes PIANO, SPEECH and BARKING. This figure shows first that potential classification errors can obviously occur: at time $t = 7$, the BARKING output probability reaches about 98% while a piano sound is perceived by the robot. Additionally, the data show the temporal dynamic audio experts can exhibit: while the piano starts playing at time $t = 3$, the

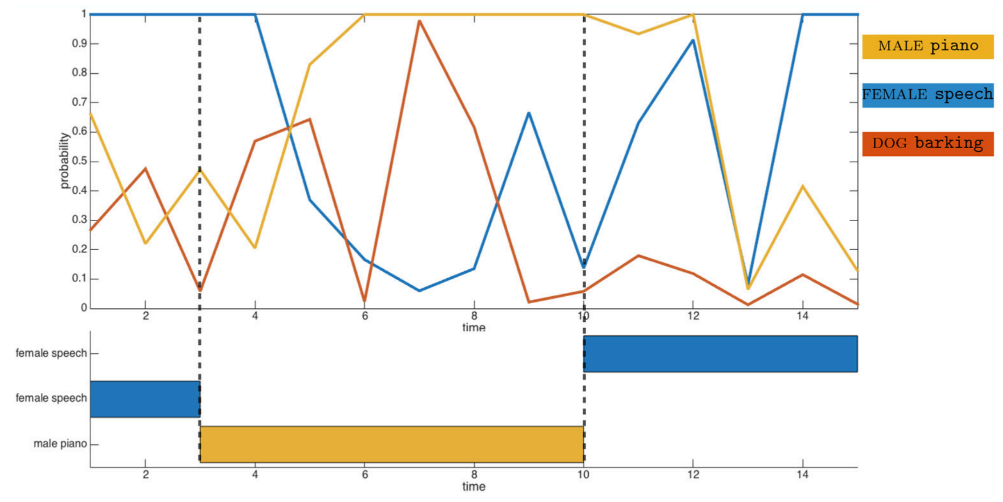


FIGURE 2 | Illustration of the audio classification experts on real perceived data. **(Top)** Probabilities of the frames to belong to the corresponding audio classes. **(Bottom)** Time description of the audiovisual objects appearance.

corresponding audio expert becomes dominant a few frames later only. This delay observed experimentally will justify later technical implementation specifics.

At this point, the notion of object already constitutes more than just a structure of data. In particular, the objects created by the HTM system embed a short-term temporal smoothing of the data $\mathbf{P}(o_j)$ they are associated with, as

$$\mathbf{P}(o_j)[t] = \frac{1}{N_t} \sum_{n=t-N_t}^{n=t} \mathbf{P}(o_j)[n], \qquad (3)$$

where $N_t \leq 10$ is the number of frames during which data $\mathbf{P}$ have been associated to o_j. This temporal smoothing enables the robot to take into account its past experience of the audiovisual data the robot perceived and that have been associated with this object, but also to lower the impact of the early potential erroneous outputs from the classification experts. Indeed, experiments have shown that most of them are prone to making more errors during the very first frames of perceived events. Thus, it is one of the goal of the HTM system to make the object identical to the event, even in the presence of classification errors. In all the following, the internal representation $e^{(l)}$ of the l-th environment being explored by the system is defined as the collection of the $N^{(l)}$ objects inside, i.e., $e^{(l)} = \{o_1, \ldots, o_{N^{(l)}}\}$. Note that $N^{(l)}$ evolves along time all along the agent life on the basis of the perceived data. Importantly, this definition of an environment—which will be augmented later on in section 3.2.1)—aims at making the difference between the topological and the semantic definition of an environment (see section 1). While the robot, through its navigation system, gets to know when a new topological environment is being explored, the HTM analyzes its audiovisual content in order to assess whether this environment is really new or if it similar to a previously explored one (as explained in section 3.2.1). In that case, this audiovisual similarity enables the robot to apply previously self-created behavioral rules making its reaction abilities way quicker. Then, one can define audiovisual categories $\mathcal{C}^{(l)}(\mathrm{c}_i^a, \mathrm{c}_k^v)$ of this l-th representation with

$$\mathcal{C}^{(l)}(\mathrm{c}_i^a, \mathrm{c}_k^v) = \{o_j \in e^{(l)}, \widehat{\mathrm{c}}^a(o_j) = \mathrm{c}_i^a \text{ and } \widehat{\mathrm{c}}^v(o_j) = \mathrm{c}_k^v\}. \qquad (4)$$

Once the events have been interpreted as objects within the internal representation $e^{(l)}$ of the robot, the HTM system analyses them through the notion of *Congruence*, described in the next section.

3. THE HEAD TURNING MODULATION SYSTEM

The *Head Turning Modulation* system is an attempt to provide a binaural and binocular humanoid robot with the ability to learn by its own how to react to unpredictable events and to consequently trigger or inhibit head movements toward them. Moreover, the system is endowed with a module that provides a multimodal internal representation of the world through a real-time learning paradigm that has no access to any prior knowledge about the environments to be explored. This system, partially introduced by the authors in Cohen-Lhyver et al. (2015), Cohen-Lhyver et al. (2016), and Cohen-Lhyver (2017) is defined as a model of attention supported by an object-based representation of the world. This section will thus present separately the two constitutive modules of the HTM system. An evaluation of each of them will be presented in simulated conditions, while the evaluation of the whole system, made in real conditions, will be presented in section 4.

3.1. Architecture of the Proposed System

The overall architecture of the HTM system is depicted in **Figure 3**. It exhibits two modules inside, each of them being dedicated to one specific task. As outlined in section 2.2, the HTM inputs are made of audio and visual classifiers outputs,

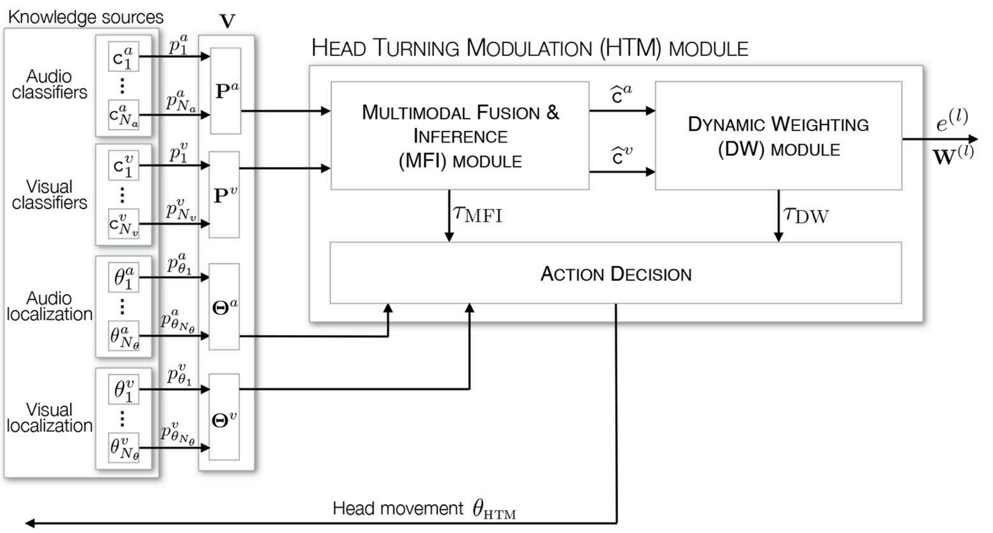


FIGURE 3 | Architecture of the HTM system. It is made of two modules, dedicated to the estimation of the audiovisual class of an event and to the computation of its importance. An additional element is in charge with the respective motor orders integration and decision.

which are used by the first module—the Multimodal Fusion and Inference (MFI) module—to provide an estimation of the audiovisual class $\widehat{c}(o_j) = \{\widehat{c}^a(o_j), \widehat{c}^v(o_j)\}$ of the currently analyzed frame. As will be shown later, such an estimation is made possible by a top-town motor feedback that allows the system to gather additional audio and visual data. On the basis on the frame estimated classification, a second module—the Dynamic Weighting (DW) module—is in charge of deciding if the currently emitting object is of interest through the computation of its congruence to the current environment. As a result, this module also exploits the motor feedback to modulate the robot attention. Since both modules require motor actions for their operations, a supplemental element is in charge with prioritizing them, depending on their respective motor activities τ_{DW} and τ_{MFI}, see **Figure 3**. Motor decisions are taken by using the localization experts providing an estimated angle of the processed event. Finally, the overall HTM system outputs a list of interpreted objects, i.e., an internal representation of the explored environment, which can be used by other KS in the TWO!EARS architecture for other tasks (modulating the exploration depending on the objects in the environment, deciding which object is of particular interest in the current scenario on the basis on the DW module module conclusions, exploiting the top-down architecture to refine the peripheral processing steps, etc.). All of these modules are introduced in the next subsections together with some intermediate illustrations and evaluations of their functioning.

3.2. The Dynamic Weighting Module

The *Dynamic Weighting* module (DW module) is the attentional part of the HTM system aiming at giving the robot an hypothesis about a possible relevant audiovisual object that would present an interest to it, in the scope of the exploration of unknown environments. As already stated, this interest is formalized through the new notion of *Congruence*, thereafter detailed.

3.2.1. Congruence: Definition and Formalization

Congruence is a notion that defines the relationship between an audiovisual event to the environment it is occurring in. It has to be brought next to the well-known and studied notion of Saliency (Treisman and Gelade, 1980; Nothdurft, 2006; Duangudom and Anderson, 2007) that describes how the perceived characteristics of a stimulus exhibit continuity, or not, with its direct surrounding. Whereas saliency is based on low-level characteristics of the signals (such as intensity, frequencies, pitch, color, contrast, etc.), Congruence relies on a higher representation of the audio and visual signals, namely the audiovisual class they belong to (see section 2.2). Congruence is thus defined as a *semantic saliency* for it relies on an already interpreted representation of the perceived data. Since the robot does not have any prior knowledge about the possible likelihood of an audiovisual event to occur in an environment, the DW module will only base its analysis on a posteriori probabilities, that is computing statistics only on what has been observed so far by the system, environment by environment. This probability of an object o_j to belong to a certain audiovisual category $\mathcal{C}^{(l)}(c_i^a, c_k^v)$ is thus defined as

$$p\left(o_j \in \mathcal{C}^{(l)}(c_i^a, c_k^v)\right) = p\left(\mathcal{C}^{(l)}(c_i^a, c_k^v)\right) = \frac{|\mathcal{C}^{(l)}(c_i^a, c_k^v)|}{N^{(l)}}, \quad (5)$$

where $|\mathcal{C}^{(l)}(c_i^a, c_k^v)|$ depicts the number of objects that have already been associated to the audiovisual category $\mathcal{C}^{(l)}(c_i^a, c_k^v)$ (as a reminder, $N^{(l)}$ corresponds to the number of objects detected so far in the l-th environment). Still following the fact that no prior knowledge is available for the robot, the system will compare this a posteriori probability to a threshold $K^{(l)} = 1/N_{\mathcal{C}^{(l)}}$ defined as the equiprobability of an object to belong to any of the categories detected so far, where $N_{\mathcal{C}^{(l)}}$ is the number of different audiovisual categories detected in the l-th environment. Such criterion has been chosen so that minimal bias is introduced in order not to promote any audiovisual category. The criterion $K^{(l)}$ evolves through time: the more audiovisual classes observed, the lower the criterion. Finally, the congruence decision follows:

$$o_j \in \mathcal{C}^{(l)}(\text{c}_i^a, \text{c}_k^v) \text{ is incongruent} \Leftrightarrow p(\mathcal{C}^{(l)}(\text{c}_i^a, \text{c}_k^v)) \leq K^{(l)}. \quad (6)$$

All the "status of congruence", that is whether they are congruent or not, of the audiovisual categories detected by the system in a given environment are then gathered into a binary vector $\mathbf{W}^{(l)} = \{p(\mathcal{C}^{(l)}(\text{c}_i^a, \text{c}_k^v)) \leq_{0,1} K^{(l)}\}, \forall(i,k)$, with $\leq_{0,1}$ a binary comparison operator. This vector of size $|\mathcal{C}^{(l)}|$ completes the definition of environments as they become collections of objects $e^{(l)}$ coupled with their congruence status $\mathbf{W}^{(l)}$. In consequence, an audiovisual class can be incongruent in an environment, but congruent in another. Since the robot would explore unknown environments during its whole life, the knowledge gained from previous explorations has to be reusable for it might speed up the exploration of new ones. Following a rule of strict inclusion of the sets of categories observed in every environment explored so far by the robot, if the set of categories detected during the exploration of an environment $e^{(i)}$ has already been observed in a previous environment $e^{(j)}$, then $\mathbf{W}^{(i)} = \mathbf{W}^{(j)}$. This redefinition of an environment implies that there is one instantiation of the DW module per environment. In addition, even in the case where there has been a reuse of information, the rules of Congruence are still computed as if the current environment was a completely new one. Consequently, if $e^{(j)}$ gets to differ at a point in time from $e^{(i)}$ and that there is no other correspondence with other environments, the $\mathbf{W}^{(j)}$ vector computed in parallel from the beginning of the exploration of $e^{(j)}$ will be from now on applied.

3.2.2. Motor Orders

Based on the congruence of all the objects, an active behavior is defined: if an object o_j is incongruent according to Equation (6), then it is worth focusing on it. A head movement can consequently be triggered in the direction of this object. At the opposite, if $p(\mathcal{C}^{(l)}(\text{c}_i^a, \text{c}_k^v)) > K^{(l)}$ the robot would inhibit this movement. But such a binary motor decision has several drawbacks, as demonstrated in Cohen-Lhyver et al. (2015). Among others, it presents a high sensitivity to classification errors, leading to erroneous motor decisions. Introducing a temporal weighting w_{o_j} of each object o_j, inspired by the temporal dynamic of the Mismatch Negativity phenomenon (Näätänen et al., 1978), filters out efficiently most of these errors. These weights are computed thanks to two different functions, depending upon the probability $p(\mathcal{C}^{(l)}(\text{c}_i^a, \text{c}_k^v))$, along

$$w_{o_j}[n] = \begin{cases} f_\omega^\bullet[n] = 1/(1+100\,e^{-2n}) & \text{if } p\left(\mathcal{C}^{(l)}(\text{c}_i^a, \text{c}_k^v)\right) \leq K^{(l)}, \\ f_\omega^\circ[n] = (1/1+0.01\,e^{2n}) - 1 & \text{else,} \end{cases} \quad (7)$$

where $f_\omega^\bullet[n]$ and $f_\omega^\circ[n]$ are increasing positive and decreasing negative functions dedicated to the weighting of incongruent and congruent objects, respectively, and n a time index. Note that n is systematically reset to 0 whenever the congruence status of the object o_j switches. From these weights, it is possible to decide which object has to be focused on. Such a decision is implemented through an adaptation of the GPR model (Gurney et al., 2001a,b) of the basal ganglia-thalamus-cortex loop involved in the motor order decision in humans. According to this model, all possible motor actions are expressed as channels of information which are by default inhibited by several afferent external connections. Depending on the goal or on the perceived stimuli, one of the channels is excited, thus promoting the motor action it is representing. Inspired by this functioning, all the objects perceived by the robot are similarly represented as information channels having a dedicated activity $\tau_{\text{DW}}(o_j)$. The vector of canal activities $\boldsymbol{\tau}_{\text{DW}}$ can be then defined as

$$\boldsymbol{\tau}_{\text{DW}} = \left(\tau_{\text{DW}}(o_1), \ldots, \tau_{\text{DW}}(o_{N_l})\right), \quad \text{with } \tau_{\text{DW}}(o_j) = -\frac{p\left(\mathcal{C}^{(l)}(\text{c}_i^a, \text{c}_k^v)\right)}{K^{(l)}}. \quad (8)$$

Thus, the higher the weight w_{o_j} of an object, the lowest the activity of its corresponding canal. The angle $\widehat{\theta}(o_j)$ estimated by the audio localization expert corresponding to the canal with the lowest activity will then be selected as the winning motor order θ_{DW}, i.e.,

$$\theta_{\text{DW}} = \widehat{\theta}(o_j), \quad \text{with } j = \arg\min_l \left(\tau_{\text{DW}}(o_l)\right). \quad (9)$$

If two different objects o_j and o_l have the same weight $w_{o_j} = w_{o_l}$, then their corresponding channels $\tau_{\text{DW}}(o_j)$ and $\tau_{\text{DW}}(o_l)$ have the same value. In such a case, the most recent object in the representation is promoted, thus introducing a motivation by novelty (Huang and Weng, 2002, 2004) (see also Walther et al., 2005). Then, Equation (8) is slightly modified by introducing a weight which is minimized for recently appeared objects, i.e.,

$$\tau_{\text{DW}}(o_j) = -\frac{p\left(\mathcal{C}^{(l)}(\text{c}_i^a, \text{c}_k^v)\right)}{K^{(l)}} \times \frac{1}{\Delta_t(o_j)}, \quad \text{with } \Delta_t(o_j) = t - t_{\text{emit}}(o_j), \quad (10)$$

where $\Delta_t(o_j)$ represents the time elapsed between the object appearance $t_{\text{emit}}(o_j)$ (reset to t when the object starts emitting again after having stopped previously) and current frame t. Note that the temporal smoothing introduced by Equation (3) does not influence the global reactivity to unexpected events, for the dynamics of the smoothing has the same order of magnitude to the dynamic of the weighting function in Equation (7).

3.2.3. Simulations and Evaluation of the DW Module

The DW module aims at controlling the head movements of an exploratory robot through the notion of Congruence of perceived audiovisual objects. Thus, what is expected from the DW module is to either trigger movements toward important audiovisual sources, and to also be able to inhibit them when necessary. To illustrate this, simulations have been conducted on the basis of the TWO!EARS architecture. Importantly, twelve audio classifiers and ten visual classifiers are actually implemented inside the software (Two!Ears, 2016a), making evaluation scenarios quite

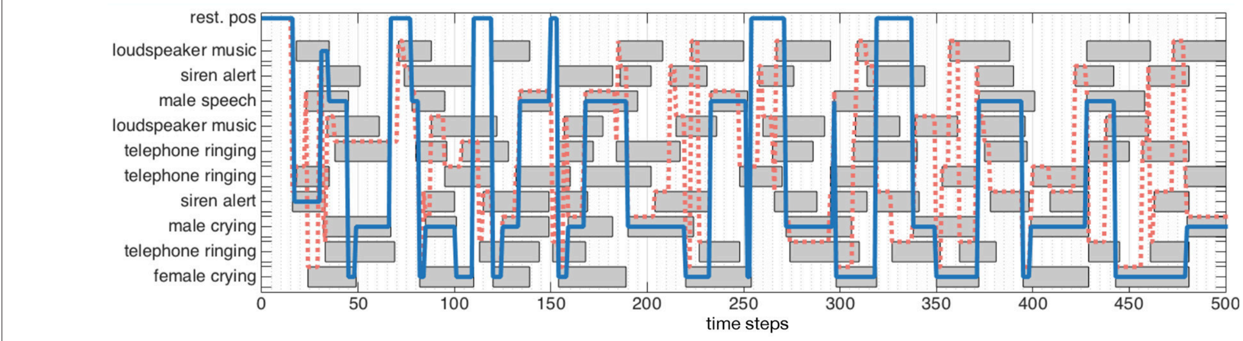


FIGURE 4 | Audiovisual sources toward which a head movement has been triggered (blue line) by the DW module, (dotted red line) by the naive robot. (Gray boxes) audiovisual sources emitting sound. A source is focused on when the lines crosses the corresponding box.

limited. Thus, instead of simulating raw (audio and visual) data used by real classifiers, their outputs $p_i^a[t]$ and $p_k^v[t]$ are rather simulated. Nevertheless, real conditions will be used later to evaluate the overall HTM system in section 4. Note that the forthcoming simulated localization experts are designed to provide the exact object audio and visual localization, the focus being put here on the congruence analysis performed by the DW module.

3.2.3.1. Simulations

Multiple evaluation scenarios are proposed, each of them being described by the number $n_{\mathcal{S}}$ of different sources in the simulated environment, the description of their azimuthal localization, their temporal appearance and disappearance, and their ground truth audiovisual classes $\mathrm{c}(\Psi)$–obviously, the HTM system does not have access to any of these. The scenarios are also defined by the maximal number of simultaneously emitting sound sources n_{sim}^{max}. While this number never exceeds five in real extreme experimental conditions, the simulations allow to incorporate up to ten audiovisual sources. At every time step t of a simulation, a vector $\mathbf{P}[t] = (\mathbf{P}^a[t], \mathbf{P}^v[t])$, from Equation (1) is sent to the HTM system. In the scope of the sole DW module evaluation, the estimated audio and visual classes of an event is directly obtained from $\mathbf{P}[t]$, i.e., on the KS outputs, according to a maximum a posteriori (MAP) estimation, with

$$\widehat{\mathrm{c}}^a_{\mathrm{MAP}} = \mathrm{c}^a_i, \quad i = \arg\max_l(p_l^a) \text{ and } \widehat{\mathrm{c}}^v_{\mathrm{MAP}} = \mathrm{c}^v_k,\ k = \arg\max_l(p_l^v). \tag{11}$$

Note that this audiovisual class estimation will be later provided by the MFI module introduced in section 3.3, as shown in **Figure 3**. However, because of the inevitable presence of classification errors, the corresponding audio and/or visual classes can be wrong (see **Figure 2**). It has been simulated through the implementation of an error rate $\varepsilon_{\mathcal{P}} \in [0, 100]\%$. At time t, a ground truth probability vector corresponding to the simulated event is generated. With respect to $\varepsilon_{\mathcal{P}}$, a "wrong" classification expert index is randomly selected by drawing its value from a uniform pseudorandom number generator. Then, its associated probability is set to be the maximal value of the whole vectors $\mathbf{P}[t]$. In the end, this will allow to judge the robustness of the approach to such classification errors.

Like proposed in Girard et al. (2002), the performance of the system is partially evaluated in comparison with a virtual "naive robot" noted $\mathfrak{R}_n$. In particular, $\mathfrak{R}_n$ will systematically turn its head toward any audiovisual source occurring in the environment, independently of its importance. For now, the simulations are made with an important restriction (explained and justified later): all the sources are in the field of view of the robot, i.e., the robot always has access to visual data.

3.2.3.2. Evaluation 1: head movements modulation by the DW module

A rather complex environment is used in the following to illustrate the functioning of the DW module: $n_{\mathcal{S}} = 10$ audiovisual sources are present with a maximum of $n_{sim}^{max} = 7$ simultaneously emitting sources. At first, let's focus on the ability of the DW module to modulate head movements by selecting only the sources of importance through the congruence analysis. Here will only be assessed the behavioral role conferred by the DW module to the robot; in consequence the simulated classification experts will be set as outputting perfect data, that is $\varepsilon_{\mathcal{P}} = 0$ (evaluations with higher error rates are made later in the paper).

Figure 4 exhibits one simulated environment, made of sources (represented as gray boxes) emitting sound along time (horizontal axis). Each source belongs to an audiovisual category represented on the left axis. Some sources might have the same audiovisual category: for instance, in this simulated scenario, the environment is made of three different telephones ringing. In addition to this "objects along time" description of the scene, **Figure 4** shows two different lines: both "pass" through objects, indicating that the robot has decided to focus on them. The blue line corresponds to the decision taken by the DW module, while the red dashed one corresponds to $\mathfrak{R}_n$. Simulations show that the DW module considers the audiovisual classes (RINGING, `telephone`) and (MUSIC, `loudspeaker`) as congruent in less than 100 time steps. This is because of their distribution with respect to the other categories: in the beginning of the simulation, objects belonging to these two categories are often present, making them less important. Consequently, the robot

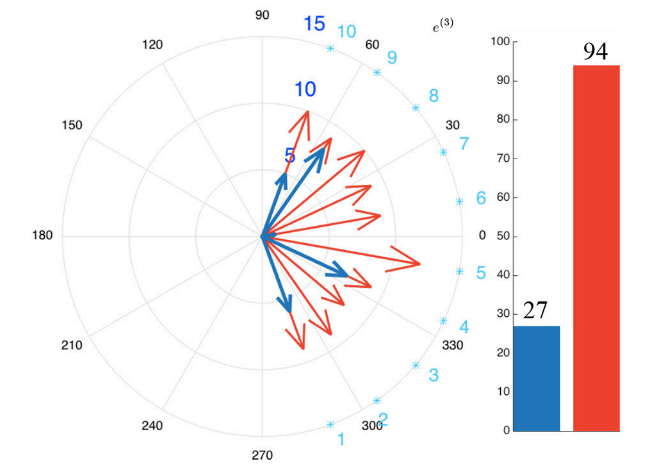


FIGURE 5 | Head movements triggered by (blue) the DW module, (red) the virtual naive robot. Every arrow points toward the position of a source and their lengths depict the number of movements toward every pointed source. (Bars:) number of movements triggered by the the MFI module (blue) and the virtual naive robot (red). The light blue numbers correspond to the position of the audiovisual sources.

will not turn its head toward those sources: there is no (motor) attentional reaction anymore. On the other hand, the categories (ALERT, `siren`), (SPEECH, `male`), (CRYING, `female`), and (CRYING, `male`) are considered as incongruent, thus requiring the robot to focus on them. Importantly, the actual meaning of those sources is not used here to decide of a reaction: one could have trade the congruent categories with the incongruent ones without any change in the global reaction. Only the frequency of apparition defined in Equation (5) is taken into account to decide the importance of a source.

In comparison, the naive robot $\mathfrak{R}_n$ turns its head every time a source starts to emit sound: it is particularly noticeable between $t = 200$ and $t = 250$ where a lot of movements can be observed. The comparison between the two behaviors is highlighted in **Figure 5**, where is depicted the total number of head movements triggered to the audiovisual sources in the environment for the DW module (blue) and naive robot (red). It appears that a drastic modulation of the exploratory behavior is obtained: using the DW module conducts to a reduction of 71.3% of the number of head movements in comparison with the naive robot. Furthermore, the DW module only triggers movements toward five sources, instead of ten for the naive robot, thus showing how Congruence—even with its simple and intuitive definition—can provide an efficient filter for the attentional behavior of the robot. Importantly, such a modulation allows the robot to use head movements, and more generally its exploratory actions, for other unrelated tasks. As long as no incongruent source is detected, head movements are free to be used for anything else. But as soon as an incongruent source pops up in the environment, the DW module will drive the head toward this source: the robot then puts its attention on it. In the end, this simple illustration shows how important it is to be able to inhibit or trigger head movements.

3.2.4. Conclusion and Limitations

The DW module is a crucial part of the HTM system in charge with providing a semantic understanding of the unknown environments the robot is supposed to explore. One of the cornerstone of this module is to be able to work without prior knowledge about the potential distribution of the audiovisual sources occurring in these environments. Thus, the DW module has to create congruence rules on the sole basis of what the robot sees and hears, that is the audio and visual labels the classification experts output. The behavior rules created are, firstly, adaptive enough to always take into account new information, since the congruence status of all the objects are computed every time a new object is detected in the environment; and secondly, broad enough to limit any bias possible in the interpretation of the perceived information: an audiovisual class can be incongruent in an environment but congruent in another one, as will be illustrated in section 4. Moreover, by not creating any prior behavioral rules (such as *if-else* statements) and by letting the system continuously being sensitive to new information, the DW module provides the robot with a life-long learning of the environments composing the world it is living in. However, one important limitation appears here: the DW module needs to have access to a *complete* audiovisual information in order to compute the congruence of any object appearing in the scene. Indeed, in the situation where a source is placed behind the robot, it would have to first turn its head toward it in order to get the full audiovisual data, to then be able to take a decision on whether or not a head movement is necessary…which is what can be called a *deadlock* situation. This is why the previous illustration of the DW module has used a setup where all the visual data were always perceivable to the robot. Obviously, this is not a realistic context at all. This is where the second module of the HTM system comes into play.

3.3. The Multimodal Fusion and Inference Module

The *Multimodal Fusion & Inference* module (MFI module) is in charge of providing the DW module with a complete information about the audiovisual sources, even when they are placed behind the robot. Moreover, the MFI module is able to cope with classification errors, i.e., to provide a stable and reliable estimation of the audiovisual classes of an object. This module is based on an online self-supervised active learning paradigm that enables the overall system to create knowledge about the audiovisual classes that are present in the environments the robot is exploring. Basically, the idea is to exploit head movements to learn the relationship between the audio and visual classes of the sources, making the robot becoming afterwards able to infer a missing modality. To begin with, the learning paradigm of the MFI module is described in a first subsection. Then, the way motor orders are triggered to learn the association between audio and visual classes is presented. An illustration of the MFI module functioning together with new details concerning the simulations, are then provided. A short discussion ends this MFI module presentation.

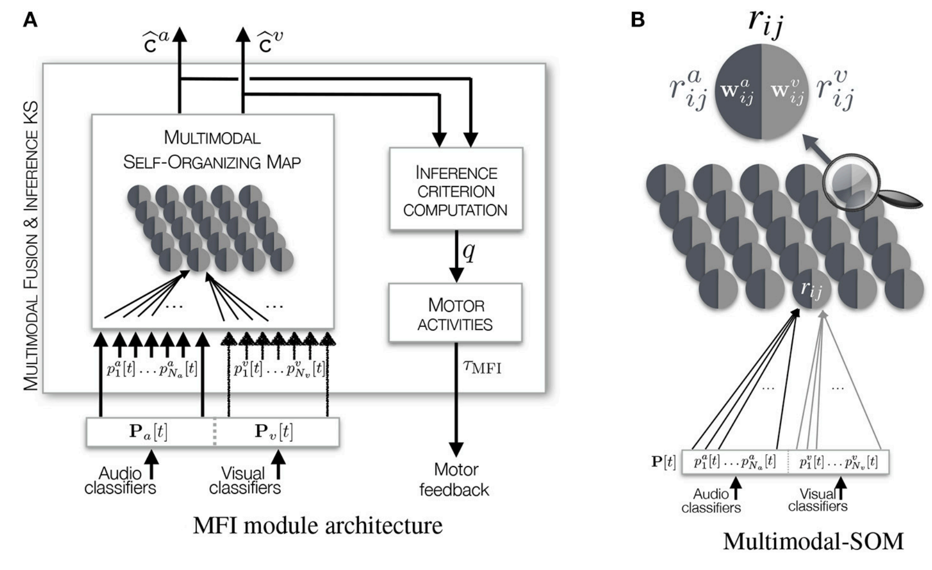


FIGURE 6 | Illustration of **(A)** the Multimodal Fusion and Inference module, **(B)** the Multimodal Self- Organizing Map. The M-SOM embeds one SOM per modality used for the definition of an object (audio and vision in our case). The representation here depicts the two subnetworks as a map containing neurons split in two parts defined by their own weights vectors, one part being dedicated to the mapping of audio data, the other to visual data.

3.3.1. The Multimodal Self-Organizing Map

The MFI module is based on a Self-Organizing Map (SOM) Kohonen (1982) which is a learning algorithm relying upon a low dimensional map on which is performed a vector quantization of a high dimensional input matrix of data, while allowing its categorization. The input data are here made of classification experts outputs gathered in the vector $\mathbf{P}[t]$, see **Figure 6A** and Equation (1). However, the traditional SOM algorithm shows one important limitation: it is unable to cope with missing data. In the case where an event originates from behind the robot, visual classifier outputs will not be relevant: the visual modality is missing. Then, two options can be chosen: (i) remove the corresponding visual components of $\mathbf{P}[t]$, or (ii) set the corresponding components to the same arbitrary value. In the former case, this would imply a change in the data dimensionality. In the latter case, this would create arbitrary meaningful data which would be misinterpreted by the SOM. Then, these two options do not offer any solution to missing data inference. This is why it is proposed to transform a classical SOM into a *Multimodal*-SOM in order to keep what makes it powerful and usable with the constraints listed before. Interestingly, Papliński and Gustafsson (2005) have developed a bio-inspired system of interconnected SOMs allowing the learning of complex multimodal data for classification purpose. But while this system possesses interesting multimodal classification properties, it lacks the essential capability of inferring missing information. More recently, Bauer and Wermter (2013) and Schillaci et al. (2014) have proposed original models based on the SOM paradigm. But while they allow the multimodal learning of perceptual data in an unsupervised way, their major drawbacks reside either in their need of significant amount of data or in the time required to converge to a stable representation of the processed data.

3.3.1.1. The subnetworks

Lets recall that a SOM is a map composed of $I \times J$ interconnected r_{ij} nodes, or neurons. The proposed modification of the original SOM consists in creating one SOM per modality, as shown in **Figure 6B**. Thus, the M-SOM is made of two (interdependent) maps, also composed by $I \times J$ interconnected $r_{ij}^{a/v}$ nodes, of size $\lceil\sqrt{N_a \times N_v}\rceil \times \lceil\sqrt{N_a \times N_v}\rceil$ (where a/v stands for *audio or visual* in a compact notation). This size has been selected to ensure that there will be at least one node available per possible audiovisual class combination, given that no prior information is available about the plausible audiovisual classes the robot will perceive during its life-long exploration. To each node is associated (i) a weights vector $\mathbf{w}_{ij}^a = (w_{ij}^a(1), \dots, w_{ij}^a(N_a))$ of size N_a for the audio subnetwork, and a weights vector $\mathbf{w}_{ij}^v = (w_{ij}^v(1), \dots, w_{ij}^v(N_v))$ of size N_v for the visual one, (ii) a (i, j) position in the map, and (iii) connections $\chi_{(ij)\to(kl)}$ between the $r_{ij}^{a/v}$ nodes and their neighbors in the same map, where $[i, k] \in [1, I]$ and $[j, l] \in [1, J]$ (with an exception for the nodes located at the edges of the map where the connectivity is reduced). The weights vectors $\mathbf{w}_{ij}^{a/v}$ associated to all the $r_{ij}^{a/v}$ nodes will become, through the iterative learning phase, the representatives of the different kinds of vectors constituting the input matrix, and thus, of the different audiovisual classes the input data capture.

3.3.1.2. Weights update

Traditionally, at every iteration n_{it} of the original SOM algorithm (the total number of iterations classically going from thirty to thousands, given the complexity of the data to be processed), the input matrix is parsed randomly until every vector has been processed once (Kohonen, 2013). For every vector explored the algorithm looks then for the closest weights vector $\mathbf{w}_{ij}$

associated to the node r_{ij} to the current input vector, in terms of their Euclidean distance. The winning neuron, that is the one presenting the closest distance to the input vector, is called the *Best Matching Unit* (BMU). It will be the location in the map where the propagation of the resemblance between the input vector and the weights vector $\mathbf{w}_{\text{BMU}}$ will start. This propagation follows a Gaussian neighborhood function $h_{ij}[n_{\text{it}}]$ (see Equation 14) of variance $\sigma[n_{\text{it}}]$ that defines the spread of the propagation. The neighborhood function is modulated by a factor $\alpha[n_{\text{it}}]$, the learning rate, making the learning powerful in the first iterations but almost non-existent in the last ones. Spreading the resemblance to the BMU's neighbors has two effects: (i) lowering the distance between the BMU and the input vector so that this neuron becomes more and more the representative of the information coded by this vector, and (ii) partially shaping the map around the BMU so that the closest to the BMU in terms of distance, the closest also in terms of information coded by the input vector. This leads to an self-organized map where regions have emerged, regions that code for similar categories. Once every vector of the matrix has been explored, a new iteration of learning starts. At every iteration n_{it} is incremented making $\alpha[n_{\text{it}}]$ and $\sigma[n_{\text{it}}]$ both decrease. Such decreasing leads to the following behavior of the learning process: at start, the propagation spreads largely in the SOM and the learning rate is at its highest; at the end of the learning, the propagation barely spreads around the BMU and the learning rate is at its lowest.

Within the M-SOM however, several changes of the traditional algorithm have been performed, changes that impact the way weights are updated. First, an audiovisual BMU r^{av}_{BMU} is now computed as the combination of the two (audio and visual) subnetworks, according to

$$r^{av}_{\text{BMU}} = r_{IJ}, \quad \text{with } (I, J) = \arg\min_{i,j} \left(\|\mathbf{P}^a - \mathbf{w}^a_{ij}\| \times \|\mathbf{P}^v - \mathbf{w}^v_{ij}\| \right), \tag{12}$$

where $\|.\|$ depicts the Euclidean distance between the vectors. This combined audiovisual BMU is associated to the combined weights vector $\mathbf{w}^{av}_{\text{BMU}} = (\mathbf{w}^a_{\text{BMU}}, \mathbf{w}^v_{\text{BMU}})$.

Secondly, the HTM does not have access to the whole matrix of data: the robot gets one vector at a time, every time a frame is analyzed by the set of KSs in the architecture. Thus, the iterative process has been revisited accordingly to this online paradigm. At every time step, the M-SOM will perform only 1 iteration of learning with the current vector (that is, there is no infinite memory of the past perceived data). However, the key principle of augmenting the resemblance between the BMU and the current vector, together with its spread, must be kept in order to reach an organized map. Taking also into account the fact that the audio classification experts from TWO!EARS get more and more precise the longer they gather data from a same sound source, the evolution of $\alpha[n_{it}]$ and $\sigma[n_{it}]$ has been reversed. The first steps of learning correspond to the minimum values of the learning rate and the variance of the neighborhood function, so that less importance is put to the very first classification experts data, and more to the next ones, following also the definition of an object (see section 2.2). Thirdly, still from the fact that the system does not have access to the whole data to be processed, it is necessary to adapt how the algorithm converges. Since the robot will always get to explore new environments during its life, there is no priorly known solutions to this learning problem. Consequently, instead of trying to reach a global convergence of the overall M-SOM, the MFI implements a *local consistency* (Chapelle et al., 2002; Zhou et al., 2004) at the audiovisual-class level (see also section 3.3.1.4). This local consistency enables the M-SOM to judge by itself whenever the learning of a particular class can be stopped or has to be continued. Thus, the value of the iteration n_{it}, that will have an impact on the values of α and σ, will be computed *object by object*: every object has its own iteration value corresponding to a certain degree in the learning process of the audiovisual class it belongs to. The choice of implementing an iteration index object by object instead of class by class, which would seem more logical, comes also from the potentially erroneous behavior of the classification experts during the first perceived audio or visual frames associated to the objects (see section 2.2). Indeed, relying directly on these outputs could promote, on the mid- to long-term, the learning of false audiovisual classes that could hamper the learning of the correct ones. The learning iteration n_{it} is now defined by

$$\begin{aligned} n_{\text{it}}[t] &= \max\left((N_{\text{it}} - n^{o_j}_{\text{it}}[t]) + 1, 1 \right) \\ \text{with } n^{o_j}_{\text{it}}[t] &= t_{\text{init}}(o_j) + (t - t_{\text{init}}(o_j)), \end{aligned} \tag{13}$$

where $t_{\text{init}}(o_j)$ is the temporal index corresponding to the initial time the object emitted sound in the current environment, and $N_{\text{it}} = 10$ corresponds to the maximal number of iterations. The value of $N_{\text{it}} = 10$ time steps has been defined experimentally with respect to two factors: (i) a too low value would put too much importance on the very first frames detected by the classifiers for a given object, and (ii) a too high value would significantly delay the local convergence of the learning for it would also delay the moment α and σ would be high enough to make the learning actually efficient.

Once r^{av}_{BMU} is found, all the weights vectors associated with every node are then updated, as described above, and according to

$$\begin{aligned} \mathbf{w}^{a/v}_{ij}[t+1] &= \mathbf{w}^{a/v}_{ij}[t] + \alpha[n_{\text{it}}]\, h_{ij}[t, n_{\text{it}}]\, \|\mathbf{P}^{a/v}[t] - \mathbf{w}^{a/v}_{ij}[t]\|, \\ \text{with } h_{ij}[t, n_{\text{it}}] &= \exp\left(-\frac{\|r^{av}_{\text{BMU}}[t] - r_{ij}\|^2}{2\sigma[n_{\text{it}}]^2} \right), \end{aligned} \tag{14}$$

where $\alpha \in [0.02, 0.9]$ represents the increasing learning rate (first and last values from Kohonen, 1990), and $h_{i,j}[t, n_{it}] \rightarrow \mathbb{R}$ is the Gaussian neighborhood function of variance $\sigma[n_{it}]$.

3.3.1.3. Estimation of the audio and/or visual classes

Every time data $\mathbf{P}[t]$ are available from the KS, the M-SOM proposes a corresponding estimated audio and visual classes $\widehat{c}^a$ and $\widehat{c}^v$, respectively. In the case where all the data are available, then the corresponding classes can be estimated along

$$\widehat{c}^a = c^a_i, \quad i = \arg\max_l w^a_{\text{BMU}}(l), \text{ and } \widehat{c}^v = c^v_k, \; k = \arg\max_l w^v_{\text{BMU}}(l). \tag{15}$$

Thus, the audiovisual class $\widehat{c}^{\text{all}}$, estimated when all the modalities are available, is given by $\widehat{c}^{\text{all}} = \{\widehat{c}^a, \widehat{c}^v\}$. All the interest of the M-SOM is its ability to provide both audio and visual classes, even if a part the KS outputs are not available. Of course, no learning is then performed, but it is the step where the network is actually exploited for inference. In the case where, for instance, the visual data are missing (i.e., the event is out of the field of view of the robot), then:

1. audio only is exploited to determine the winning (audio) node r^a_{BMU} in the audio map, whose associated weight vector $\mathbf{w}^a_{\text{BMU}}$ can be used to estimate the audio class $\widehat{c}^a = c^a_i$, with $i = \arg\max_l w^a_{\text{BMU}}(l)$ like in Equation (15);
2. the winning (visual) node is deduced from audio by $r^v_{\text{BMU}} = r^a_{\text{BMU}}$: this is exactly where the learned interlink between audio and visual data is exploited. The corresponding visual class $\widehat{c}^v$ can then be deduced from the associated weight vector $\mathbf{w}^v_{\text{BMU}}$ along $\widehat{c}^v = c^v_k$, with $k = \arg\max_l w^v_{\text{BMU}}(l)$.

In the end, the audiovisual class $\widehat{c}^{\text{miss}}$, estimated when one modality is missing, is then given by $\widehat{c}^{\text{miss}} = \{\widehat{c}^a, \widehat{c}^v\}$. Of course all the reasoning is identical when the other modality is missing: the available data drive the missing modality for inference.

3.3.1.4. Convergence and the inference criterion

A key principle in learning algorithms is their ability to converge to one of the acceptable solutions of the problem to be solved. However in the proposed context, different environments made of possibly different audiovisual sources might be explored during the robot life. Then, it is clearly impossible to define one global good solution to the problem. Nevertheless, the proposed M-SOM possesses a characteristic of *local consistency* (see section 3.3.1.2). Within the classical SOM algorithm, convergence means that the whole map is organized such that the different nodes are grouped in meaningful entities that code part of the input data. In the proposed M-SOM, it is proposed that the algorithm always keeps a free part in the map, i.e., nodes not coding for any audio or visual classes. This would allow the network to include new audiovisual classes, discovered all along the interaction with new environments during the robot life. Looking for local consistencies, rather than reaching for global convergence, is implemented through the definition of a criterion for each audiovisual category already created, indicating how much this category has been learned so far and if its learning can be stopped. The multimodal learning performed by the MFI module is supported by head rotations to the sources to be learned. It allows to bring the visual sensors in front of them in order to learn the association between the corresponding audio and visual classes. But these head movements are no longer useful once the M-SOM has enough knowledge about the audiovisual classes, thus justifying the need to (i) inhibit these head movements, and (ii) being able to judge when this amount of knowledge is sufficient. Then, an *inference ratio* $q(\mathcal{C}^{(l)}(c^a_i, c^v_k))$ for the audiovisual category $\mathcal{C}^{(l)}(c^a_i, c^v_k)$ is defined as

$$q\left(\mathcal{C}^{(l)}(c^a_i, c^v_k)\right) = \frac{\sum_{n=1}^{n=t} \delta^{\text{miss}}_{i,k}[n-1]\, \delta^{\text{all}}_{i,k}[n]}{\sum_{n=1}^{n=t} \delta^{\text{miss}}_{i,k}[n]}, \quad \text{with } \delta^{\text{all/miss}}_{i,k} = \begin{cases} 1 \text{ if } \widehat{c}^{\text{all/miss}}(o_j) = \{c^a_i, c^v_k\}, \\ 0 \text{ else.} \end{cases} \tag{16}$$

This inference ratio is computed by comparing the number of times the audiovisual category $\mathcal{C}^{(l)}(c^a_i, c^v_k)$ has been obtained (or inferred) with one missing modality ($\delta^{(\text{miss})}_{i,k} = 1$) at time $n-1$ and confirmed at time n ($\delta^{(\text{all})}_{i,k} = 1$) by a head movement with all modalities available, with the total number of inference. Thus, $q(\mathcal{C}^{(l)}(c^a_i, c^v_k))$ captures the ability of the MFI module to infer correctly a missing modality, category by category. The inference ratio always lies between 0 and 1, where 1 means that the category has always been perfectly inferred. On this basis, $q(\mathcal{C}^{(l)}(c^a_i, c^v_k))$ is compared to a criterion $K_q \in \mathcal{R}^+ = [0, 1]$: if a modality is missing, the MFI module will attempt to infer it, and as long as the inference ratio of the corresponding audiovisual category is lower than K_q, a head movement will be triggered toward the corresponding source. Thus, the system grabs the missing information and feeds the M-SOM, which can then learn the audiovisual association. Of course, once the full audiovisual data is obtained, a comparison with the previous inference is made and the inference ratio is updated accordingly. If the inference ratio gets higher than the criterion K_q, the learning is considered as being good enough to trust the inference made by the MFI module, and inhibit consequent head movements toward the sources belonging to the corresponding audiovisual category. Remark that the criterion K_q has an influence on the behavior of the MFI module (Cohen-Lhyver, 2017). A low threshold allows a quick confidence in the inference, thus freeing head movements for other tasks, whereas a high K_q value pushes the system to be very careful about its inferences.

3.3.2. Motor Orders

As for the DW module, the MFI module is able to trigger head movements toward sources of *interest*. This interest is now formalized by the lack of confidence in the knowledge of the audiovisual category a source might belong to. As previously explained, turning the head toward a source might enable the visual sensors to get the missing visual data, thus giving to the MFI module the opportunity to learn the interlink between the audio and visual modalities, but also to eventually confirm/refute an inference. Like for the DW module, the head movements modulation is inspired by the GPR model (see section 3.2.2), but through a different expression of the activities $\tau_{\text{MFI}}(o_j)$ for the object o_j with audiovisual category $\mathcal{C}^{(l)}(c^a_i, c^v_k)$, now given by

$$\tau_{\text{MFI}}(o_j) = \frac{q(\mathcal{C}^{(l)}(c^a_i, c^v_k))}{K_q} \times \delta^{(i,k)}(n), \quad \text{with } \delta^{(i,k)}(n) = \begin{cases} -1 \text{ if } n < (t_p = 10), \\ 1 \text{ else,} \end{cases} \tag{17}$$

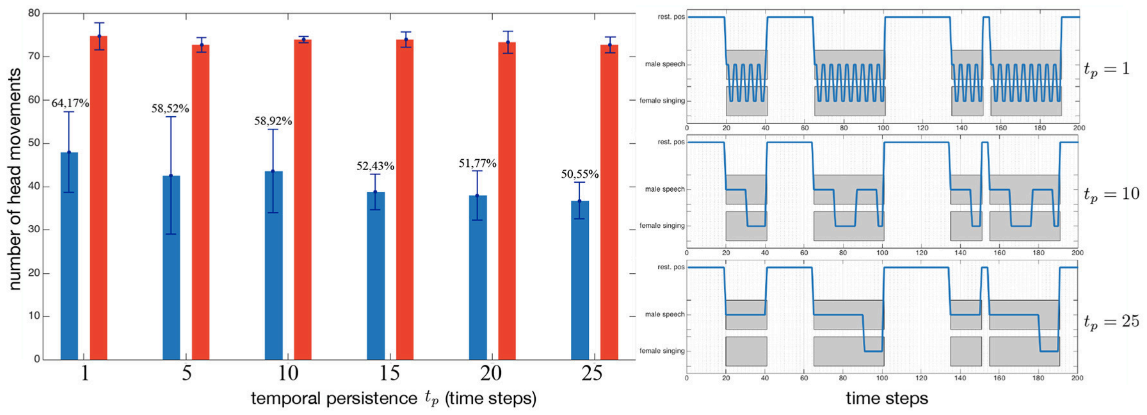


FIGURE 7 | Impact of the temporal persistence, introduced in Equation (17), (left) on the number of triggered head movements in a complex environment, and (right) the behavior of the robot in a simplified case for illustration purposes. (Blue bars:) robot driven by the MFI module, (red bars) naive robot. Percentages depict the ratio between the naive robot and the MFI module.

where $n = t - t_{\text{foc}}(o_j)$, with $t_{\text{foc}}(o_j)$ the first time the object has been focused on by the MFI module. Then, the angle $\widehat{\theta}(o_l)$ estimated by the localization expert and corresponding to the canal with the lowest activity is selected as the winning motor order θ_{MFI}, i.e.,

$$\theta_{\text{MFI}} = \widehat{\theta}(o_l), \quad \text{with } l = \arg\min_j(\tau_{\text{MFI}}(o_j)). \tag{18}$$

The term $\delta^{(i,k)}(n)$ in (17) introduces a form of temporal persistence through a positive feedback loop, as observed in the thalamus by Redgrave et al. (1999), Gurney et al. (2001a), and Meyer et al. (2005). The value of $t_p = 10$ has been set experimentally after several comparisons and evaluations. The impact of this persistence in a complex environment (eight sources with five simultaneously emitting) is illustrated in the left panel of **Figure 7**, where the blue bars depict the number of head movements triggered by the MFI module, while the red bars, by the naive robot $\mathfrak{R}_n$ (these numbers are obviously not affected by the temporal persistence applied to the MFI module). The main point is that the temporal persistence t_p constitutes only a small part of the head movements control: 13.6% less head movements between $t_p = 1$ and $t_p = 25$. The real benefits of temporal persistence is shown in **Figure 7** (right): with $t_p = 1$, the robot exhibits oscillations between two sources, potentially damaging the internal representation of the world (confusions in binaural cues computations, speed of the movement...). With $t_p = 25$, a pervert effect of a too long persistence is also shown: the system often ghosts completely the (SINGING, female) source, preventing itself from learning it.

3.3.3. Evaluation 2: Classification Rates of the mfi module

The MFI module aims providing a corrected audiovisual information from the classification experts. In order to assess the contribution brought by this module, a good audiovisual classification rate $\Gamma(o_j)[t]$ is defined by comparing the audio and visual classes associated to all the objects detected by the system with the ground truth, according to

$$\Gamma(o_j)[t] = \mathbf{a} \times \sum_{k=t_i}^{t} \gamma(o_j)[k]$$
$$\text{with } \gamma(o_j)[k] = \begin{cases} 1 \text{ if } \widehat{\mathrm{c}}(o_j)[k] = \mathrm{c}(\Psi_j)[k], \\ 0 \text{ else}, \end{cases} \tag{19}$$

with $\mathrm{c}(\Psi_j)[k]$ being the ground truth audiovisual class of the event Ψ_j captured as the object o_j at time k in the internal representation, and $\mathbf{a} = 1/[1, ..., (t - t_i) + 1]$ corresponding to the elapsed time between the first time step t_i when the MFI module provided a classification of the object o_j, and the current time t. The overall good classification rate is given by applying a sliding window on all $\Gamma(o_j)$ computed from the beginning of the exploration, along

$$\bar{\Gamma}_{\text{MFI}}[t] = \frac{1}{N^c_{obj}[t]} \sum_{j=1}^{N^c_{obj}[t]} \Gamma(o_j)[t] \tag{20}$$

with $N^c_{obj}[t]$ the number of processed objects by the MFI module at time t (this number could be inferior or equal to the total number of objects present and emitting, noted N_{obj}). In parallel, the same process is made for the naive robot $\mathfrak{R}_n$ [knowing that this one performs the fusion of the classification experts themselves through a maximum a posteriori approach, along Equation (11)], according to

$$\bar{\Gamma}_{\mathfrak{R}_n}[t] = \frac{1}{N_{obj}[t]} \sum_{j=1}^{N_{obj}[t]} \Gamma(o_j)[t]. \tag{21}$$

In addition, a measure of the classification performance of an omniscient (thus unrealistic) robot is also computed, noted $\bar{\Gamma}'_{\mathfrak{R}_n}[t]$. This robot has full access to every auditory and visual

TABLE 1 | Simulation setup for Evaluation 2.

Evaluation 2						
$e^{(l)}$	$n_{\mathcal{S}}$	n_{sim}^{max}	T	$\lvert\mathcal{C}^{(l)}\rvert$	K_q	$\varepsilon_{\mathcal{P}}$
1 to 5	3	3	1000	2	0.8	0.1, 0.3, 0.5, 0.7, 0.9
6 to 10	5	5	1000	3	0.8	0.1, 0.3, 0.5, 0.7, 0.9
11 to 15	7	7	1000	4	0.8	0.1, 0.3, 0.5, 0.7, 0.9
16 to 20	10	10	1000	6	0.8	0.1, 0.3, 0.5, 0.7, 0.9

Each setup is repeated 5 times for a total of 100 simulations.

information, even when the objects are out of the sight of the robot. The simulation setup is presented in **Table 1**. Twenty different multisource environments are simulated, each of them in possibly different conditions (number of sources, number of simultaneously emitting sources, error rates, etc.). The resulting good audiovisual classification rates are regrouped in **Table 2**, mainly organized by increasing error rates $\varepsilon_{\mathcal{P}} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

At first, let us consider the naive omniscient robot $\mathfrak{R}'_n$. As expected, it presents a mean good audiovisual classification rate $\bar{\Gamma}_{\mathfrak{R}'_n}$ almost equal to $1 - \varepsilon_{\mathcal{P}}$ for all tested conditions. In contrast, the realistic naive robot $\mathfrak{R}_n$ (having only access to the data it is able to perceive) systematically exhibits lower rates $\bar{\Gamma}_{\mathfrak{R}_n}$. Clearly, the main flaw of this robot is its incapacity to perform any inference, which turns to be a critical capability in multisource environments. In comparison, the proposed MFI module outperforms both naive robots, for almost any error rates and number of sources (except for only one case: $\varepsilon_{\mathcal{P}} = 0.1$ and $n_{\mathcal{S}} = 10$). The last column in **Table 2** exhibits the ratio between the best naive robot $\mathfrak{R}_n$ (given by $\bar{\Gamma}'_{\mathfrak{R}_n}[t = T]$) and the MFI module: the greater $\varepsilon_{\mathcal{P}}$, the higher the ratio, except with $\varepsilon_{\mathcal{P}} = 0.9$. In this case, the error rate is anyway so high that the interest in exploiting such corrupted data is almost null. However, even in very challenging conditions involving a very high $\varepsilon_{\mathcal{P}} = 0.7$ in a multisource context, the MFI module provides on average a 2.4 times better good audiovisual classification rate than with the classifier outputs.

3.3.4. Discussion

The proposed MFI module, mainly based on the M-SOM, provides an online self-supervised active learning paradigm to be able to process erroneous and/or missing data in the particular context of the exploration of unknown environments. The overall goal of the MFI module is thus to feed the DW module with correct audiovisual classes the perceived objects belong to, with respect to a very short learning time constraint (down to a few seconds only). The *active* capabilities of the MFI module is of very much importance here, for it enables the intensive use of head movements to gather, whenever it is necessary, and in real-time, additional data to refine the knowledge the module has of the world under exploration. A fundamental question arises with the problem of audio and visual classes association when considering one-to-one

TABLE 2 | Good classification rates for different error rates and different numbers of sources.

Evaluation 2: Results					
$\varepsilon_{\mathcal{P}}$	$n_{\mathcal{S}} - n_{sim}^{max}$	$\bar{\Gamma}_{\mathbf{MFI}}[t = T]$	$\bar{\Gamma}'_{\mathfrak{R}_n}[t = T]$	$\bar{\Gamma}_{\mathfrak{R}_n}[t = T]$	ratio: $\frac{\bar{\Gamma}_{\mathbf{MFI}}[t=T]}{\bar{\Gamma}'_{\mathfrak{R}_n}[t=T]}$
0.1	3 − 3	0.982 (0.027)	0.894 (0.021)	0.503 (0.073)	1.098
	5 − 5	0.988 (0.025)	0.899 (0.012)	0.339 (0.039)	1.099
	7 − 7	0.960 (0.023)	0.893 (0.016)	0.264 (0.021)	1.075
	10 − 10	0.866 (0.047)	0.887 (0.018)	0.182 (0.014)	0.976
	Mean	0.949	0.893	0.322	1.063
0.3	3 − 3	0.992 (0.020)	0.703 (0.042)	0.414 (0.055)	1.411
	5 − 5	0.987 (0.022)	0.692 (0.017)	0.265 (0.014)	1.426
	7 − 7	0.942 (0.028)	0.691 (0.014)	0.198 (0.017)	1.363
	10 − 10	0.883 (0.041)	0.689 (0.011)	0.145 (0.014)	1.281
	Mean	0.951	0.693	0.255	1.372
0.5	3 − 3	0.973 (0.026)	0.493 (0.020)	0.280 (0.031)	1.973
	5 − 5	0.965 (0.043)	0.496 (0.021)	0.189 (0.034)	1.945
	7 − 7	0.899 (0.048)	0.492 (0.018)	0.145 (0.019)	1.827
	10 − 10	0.836 (0.042)	0.492 (0.018)	0.103 (0.010)	1.699
	Mean	0.918	0.493	0.179	1.862
0.7	3 − 3	0.774 (0.087)	0.282 (0.030)	0.165 (0.028)	2.744
	5 − 5	0.737 (0.105)	0.294 (0.014)	0.120 (0.023)	2.506
	7 − 7	0.683 (0.133)	0.296 (0.016)	0.081 (0.012)	2.307
	10 − 10	0.550 (0.117)	0.293 (0.016)	0.064 (0.011)	1.877
	Mean	0.686	0.291	0.107	2.357
0.9	3 − 3	0.213 (0.060)	0.092 (0.019)	0.054 (0.019)	2.315
	5 − 5	0.152 (0.064)	0.102 (0.012)	0.039 (0.007)	1.490
	7 − 7	0.174 (0.075)	0.100 (0.009)	0.031 (0.005)	1.740
	10 − 10	0.140 (0.066)	0.100 (0.009)	0.019 (0.006)	1.400
	Mean	0.169	0.098	0.035	1.724

Every results is an average of 5 repetitions of every conditions with standard deviation in parentheses, for a total of 100 simulations. $\mathfrak{R}_n$ corresponds to the naive robot, and $\mathfrak{R}'_n$ to the unrealistic omniscient robot. Values are rounded up to the third decimal.

audiovisual pairs, i.e., that each audio label is associated with only one visual label, and vice and versa. In the evaluations presented in this section, such pairing limitation was not used: an audio label could have several visual correspondences, such as SPEAKING, `male`, SPEAKING, `female`, or SPEAKING, `child`. However, given these audiovisual labels examples, it is not possible for the MFI module to create an information that does not exist: from the audio label SPEAKING, it is impossible to determine whether the corresponding visual label is `male`, `female`, or `child`. The MFI module still outputs an hypothesis corresponding, given how the M-SOM learning algorithm works, to the most observed so far audiovisual pair. Such limitation of the MFI module only comes from the limits of the classification experts themselves: if the classifiers cannot distinguish a female voice from a male one, nor would the MFI module. Such a case will be shown and also discussed in section 4.4, when evaluating the whole system in real environments.

3.4. Conclusion

The Head Turning Modulation system is composed with two modules: the Dynamic Weighting module (DW) and the Multimodal Fusion and Inference module (MFI), each of them having been described in this section. The DW module is an attentional component, working on the sole basis of observed data in unknown environments, from which it enables the robot to turn its head to audiovisual sources considered as "of importance." Coupled to it is the MFI module that learns the relationship between the modalities that are used to define an object (audition and vision in this case). Based on a Multimodal Self-Organizing Map (M-SOM), the MFI module is able to create the knowledge required by the DW module to work properly. This knowledge consists in the fusion of multimodal data into a corrected database of audiovisual categories, knowledge that is created through online active self-supervised exploration of the audiovisual sources appearing in the unknown environments. Both modules can trigger head movements independently, and their combination necessitates an adaptation of the motor orders expressions of the modules.

The next section will present the results obtained in real environments with the real robot embedding the whole TWO!EARS software (including the integration of the HTM system), and processing real audio and visual data.

4. COMBINATION OF THE TWO MODULES

The previous section was dedicated to the individual presentation of each module constituting the HTM system, while providing limited evaluations in simulated conditions. This section is now concerned with the combination of the DW module and the MFI module together, with their evaluation in realistic conditions, i.e., on a real robot and with real audio and visual data. At first, one have to deal with the fact that theses two modules are both able to generate competitive head movements. The way they are prioritized is described in a first subsection. Next, the experimental setup is carefully described in a second subsection. Then, experimental results are provided in a third subsection, aiming at demonstrating the benefits of the overall system in the audiovisual scene understanding.

4.1. Combined Motor Orders: Evaluation 3

It has been shown in section 3 that the DW module and the MFI module both exploit head movements to better their respective operations. Trying to make them able to work together then requires a prioritization of them. On the one hand, the DW module provides the robot with potential sources to be focused on, on the basis of their computed congruence; on the other hand, the MFI module aims at estimating audiovisual classes of objects inside the environment, even with potential classification errors and lack of data. It seems then obvious to set the priority to the MFI module: having a reliable audiovisual classes estimation system is required for the attentional module to take relevant decisions. This prioritization introduces a new activity τ'_{DW} for the DW module which is now defined, for an object o_j, by

$$\begin{aligned} &\tau'_{\mathrm{DW}}(o_j) = \tau_{\mathrm{MFI}}(o_j) - \tau_{\mathrm{DW}}(o_j) \times \delta(\tau_{\mathrm{MFI}}(o_j)), \\ &\text{with } \delta(x) = \begin{cases} 1, \text{ if } x \geq 1, \\ 0, \text{ otherwise.} \end{cases} \end{aligned} \tag{22}$$

On this basis, the motor order θ_{HTM} selected to drive the head is computed along

$$\begin{aligned} &\theta_{\mathrm{HTM}} = \widehat{\theta}(o_l), \quad \text{with } l = \arg_{j_1,j_2} \min(\tau'_{\mathrm{DW}}(o_{j_1}), \tau_{\mathrm{MFI}}(o_{j_2})) \\ &\text{where } \begin{cases} j_1 = \arg\min_l(\tau'_{\mathrm{DW}}(o_l)), \\ j_2 = \arg\min_k(\tau_{\mathrm{MFI}}(o_k)), \end{cases} \end{aligned} \tag{23}$$

i.e., the object with the lowest DW module or MFI module activity is selected. Such a modification of the motor activity expression enables the MFI module to take over the lead on the DW module. The evaluation of such a modification in the motor commands decision system can be performed again in simulation, along the same procedure as in the previous simulations, see **Figure 8**. Let us consider an environment made of five objects, belonging to three different audiovisual categories. Each of these objects emit sounds along time, according to the time plot shown in **Figure 8** (bottom). **Figure 8** (top) exhibits the three-phase behavior of the motor decision algorithm. At the very beginning, only the MFI module is responsible for the head movements: the system is learning the association between audiovisual classes. Little by little, the inference provided by the MFI module does not need motor confirmation for some of the classes: the DW module can now compute congruence of the corresponding objects and potentially trigger head movements. In the end, all the audiovisual classes are correctly learned by the MFI module, letting the sole DW module in charge with head rotations. Of course, the head movements triggered by the DW module are also used to feed the M-SOM.

4.2. Experimental Setup and Data Generation

The overall system has been evaluated in a realistic environment by using a real robot integrating the whole TWO!EARS software and evolving in a real room. In practice, two different robots have been actually used: one mobile platform from LAAS-CNRS (Toulouse, France) named JIDO, the other one from ISIR (Paris, France) named ODI, see pictures in **Figure 9**. Both platforms support a KEMAR HATS (Head And Torso Simulator), whose necks have been motorized to control their head movements in azimuth (Bustamante et al., 2016). A HATS is a manikin endowed with two microphones placed inside two pinnae which mimics the acoustic effect of the head (and torso) on the left and right ear signals, thus producing a realistic binaural information, close to what a human could actually hears. The servo control of the head is ensured by a set including a motor, its gear head, an encoder, and an Harmonica electronic controller from ELMO, mounted inside the HATS. A ROS node dedicated to the head control is in charge of controlling this motorization, allowing real-time servoing of the head movements by using

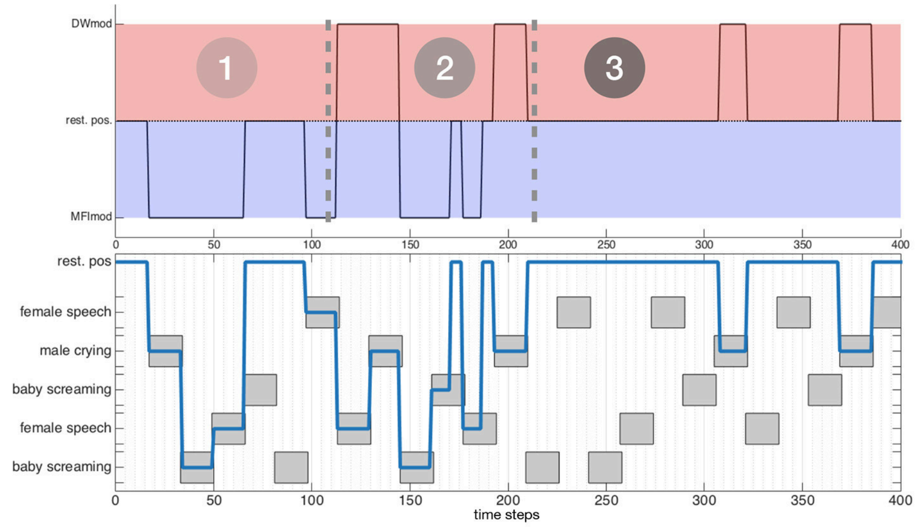


FIGURE 8 | Behavior of the combined modules in three phases. The testing environment is composed of five audiovisual sources and is willingly simple for illustration purposes. **(Top)** Module ordering the head movement. **(Bottom)** Temporal course of the exploration of the environment; gray boxes depict the temporal course of emitting sources.

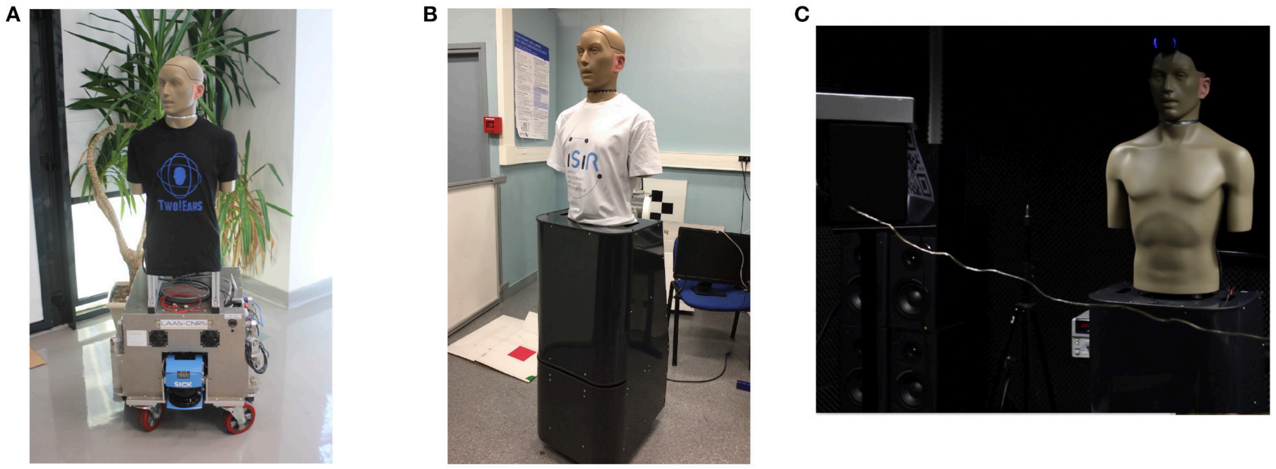


FIGURE 9 | The two robotic platforms used in the project, both supporting a motorized KEMAR HATS. ODI has been used in this paper for the HTM evaluation. **(A)** The ODI platform. **(B)** The ODI platform. **(C)** ODI, facing a loudspeaker with a QR code attached on it.

possibly different feedback control options like position or velocity setpoints. In this paper, the positions deduced from Equation (23) are directly sent to the ROS node to control the head in position. These two robots are very much alike, except for vision: the one used at ISIR for the experiments used in this paper is only endowed with monocular vision. However, as already argued, the HTM system is not dependent on the way each modality works, but only on the identification experts, be they dedicated to monocular or binocular vision for instance.

Everything related to the platform and data acquisition is handled by the ROS middleware, running directly on the robot: navigation, obstacle avoidance, image and audio captures, etc. Note that a dedicated ROS binaural processing node has been developed during the project, so that most of the audio cues required for sound localization, recognition and separation are directly computed in real-time on the robot. State-of-the-art ROS nodes dedicated to vision (acquisition and processing) have also been used. All the data computed on the robot are then transmitted to another computer running the TWO!EARS framework thanks to a MATLAB-to-ROS bridge. This bridge has been entirely designed to deal with the proposed bottom-up and top-down approach of the project, so that all the ROS nodes can be easily parameterized on the fly and in real time. Then, all the steps required for the "cognitive" analysis (i.e., object localization, recognition, fusion, etc.) runs under MATLAB.

Experiments used in this paper have been conducted in a pseudo-anechoic room populated with loudspeakers over which QR codes have been attached to, see **Figure 9C**. These are used by a ROS node to extract the visual labels of each object directly and with a recognition rate similar to the one obtained through the binocular vision of JIDO with the Line-Mod algorithm Hinterstoisser et al. (2012). All the sounds emitted from the loudspeakers belong to a database constituted of sounds used to train the audio experts in recognition. In other terms, all the sounds can be recognized by at least one expert in the architecture. Then, the HTM has been evaluated in experimental conditions by two scenarios: the first emphasizes the global behavior of the system, while the second focuses on the fusion and classification abilities of the MFI module. Whatever the scenarios, they all works the following way: sounds are emitted from one or multiple loudspeakers, possibly at the same time. Depending on how the head of the KEMAR is turned, some QR codes can be manually changed from one loudspeaker to another to simulate an object movement in the environment. The HTM system then gathers classification and localization results coming from the audio and visual experts, and triggers some head movements accordingly. A scenario is entirely described by the number of different objects in the scene and by the time description of their localization, appearance and disappearance, exactly like in the previous simulations. Of course, ground truth audio and visual classes of each object are known, thus allowing a careful evaluation of the overall system performance. Note that the audio experts used in the following experiments have been set up by using data from a database recorded in a different acoustic environment. Since they all rely on a prior learning step exploiting these data, there will be a mismatch between their learning and testing phase. The main consequence is mainly a lower frame recognition rate, evaluated to about 37% for the four classifiers used here, and that have been chosen amongst the most performing ones (Two!Ears, 2016a). The same applies to the localization algorithm, with less consequences: experiments still show a good ability to localize sounds with a precision of about 7.7° (including front-back confusion). Finally, the visual recognition of QR codes works almost perfectly, while being quite sensitive to changes in illuminations. Of course, both phenomena are dealt with the HTM system, which has been entirely designed to cope with recognition errors and lack of data, as show in the next subsections.

4.3. Evaluation 4: Global Behavior

This first evaluation aims at demonstrating how the two modules constituting the HTM system cooperate together in order the exploratory robot an additional understanding of the world. The evaluation consists in presenting to the system three successive environments made of three to four objects, as summarized in **Table 3**. The audiovisual sources of the environments are placed around the robot and emit sound intermittently, according to the time scenario shown in **Figure 10** (bottom). Exactly like in simulations, the real robot is compared to its naive counterpart $\mathfrak{R}_n$, turning its head toward every audiovisual events regardless of their meaning. To begin with, the HTM builds a first representation $e^{(1)}$. As shown in **Figure 10**, the robot starts

TABLE 3 | Experimental setup for Evaluations 4 & 5.

$e^{(i)}$	$n_{\mathcal{S}}$	n_{sim}^{max}	$c(\Psi_j)$	$\theta(\Psi_j)$	K_q
EVALUATION 4					
			dog barking n°1	320°	
1	3	1	*dog barking* n°2	35°	0.6
			female speech	70°	
			baby crying n°1	70°	
2	3	1	*baby crying* n°2	35°	0.6
			female piano	320°	
			baby crying n°1	70°	
			baby crying n°2	35°	
3	4	1	*dog barking*	320°	0.6
			male speech	280°	
EVALUATION 5					
1	5	1	*female speech*	320°	0.6
			female piano	30°	
			male speech	60°	
			dog barking	90°	
			baby screaming	280°	

by turning its head toward the first two audiovisual sources (BARKING, `dog` and SPEECH, `female`), driven by the MFI module since these audiovisual classes are brand new to it. As already outlined in the previous subsection, the HTM tries to learn the audiovisual association between these two classes. This learning is done very quickly: one can observe at time index $t = 28$ (corresponding to the "real" time 14 s) that the robot turns its head to its resting state (blue line going at the top of the figure), meaning that neither the DW module nor the MFI module requires a head movement toward the sources BARKING, `dog`: these sources are not of interest anymore, and hearing the sound BARKING is sufficient to infer the visual class `dog`. Nevertheless, one can remark a glitch in the head movement decision at $t = 30$, as the last attempt of the MFI module to learn the BARKING, `dog` audiovisual association. At $t = 41$ (20.5 s), the robot turns its head again toward the source (SPEECH, `female`): with two BARKING, `dog` for one SPEECH, `female` in $e^{(1)}$, the probability for this last audiovisual category $p(\mathcal{C}^{(1)}(\text{SPEECH}, \texttt{female})) = 1/3$ falls below $K^{(1)} = 1/2$, thus making any object of this audiovisual category incongruent. Then, the robot explores a second environment. It is similar in terms of frequencies of apparition of each audiovisual categories, even if their meaning (at least, to us) is different: the two BARKING, `dog` are trade for two CRYING, `baby`, while the category SPEECH, `female` is replaced by PIANO, `female`. Logically, the obtained behavior is similar: a quick learning of the audiovisual association allows then the head to be controlled by the DW module on the basis on congruency computations. Interestingly, the understanding of this second environment by the DW module could appear as counterintuitive in comparison with how humans might have reacted by favoring the two objects CRYING, `baby`. This more social reaction could nevertheless be handled by some additional KS from the TWO!EARS architecture which could modulate the

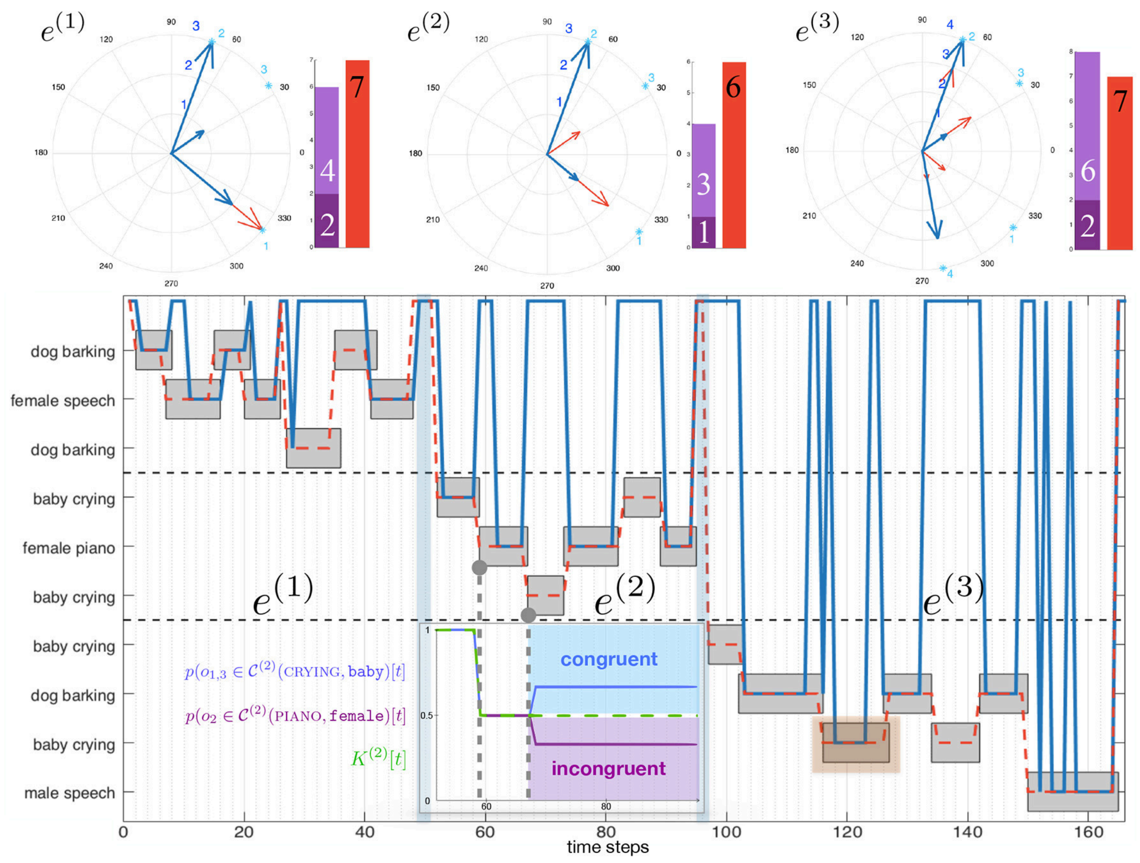


FIGURE 10 | (Top) Number of head movements triggered during the exploration of each environment by (blue) the HTMKS, (red) the virtual naive robot. Each arrow points at a source and their length represent the number of movements. The light blue numbers correspond to the position of the sources. (Purple bars:) total number of movements triggered by (dark) the MFI module, (light) the DW module (black numbers are their sum). (Red bars:) number of movements triggered by the virtual naive robot. **(Bottom)** Movements triggered by (blue line) the HTMKS, and (dotted red line) the naive robot. (Gray boxes:) temporal course of the scenarios. The semi-transparent red box at t = 116 highlights the significant wrongful discrepancy that occurred between the actual audiovisual class of the object and the perception of the HTM (error that is corrected soon after, see text for more details). Additionally, the subfigure present in the delineated box at the bottom of $e^{(2)}$ represents the evolution of $K^{(2)}$ together with the posterior probabilities of the two audiovisual classes observed in $e^{(2)}$ (in light blue for o_1 and o_3, in purple for o_2). The comparison of the all the $p(o_j)$ and $K^{(l)}$ justifies the potential triggering of head movements by the DW module, as observed at $t = 74$ and $t = 90$.

overall reaction of the robot w.r.t. the current task (Ferreira and Dias, 2014). Finally, a third environment is explored. It will allow to demonstrate the benefits of reusing information between the representation of environments, see section3.2.1. Indeed, the scene begins with a CRYING, baby which does not trigger any head movement: while being in a new environment, the HTM system considers at this point that this third environment is very likely to be the same as $e^{(2)}$ where this audiovisual class was considered as congruent. Consequently, the congruence computations of each audiovisual categories in the previous environment can still be used, and no head movements toward this now object is performed. However, as soon as a new object eliminates the possibility to be in an environment similar to $e^{(2)}$ pops up, a new representation $e^{(3)}$ is created. Thus, when the source BARKING, dog appears in the scene, a head movement is immediately triggered toward it, since it is incongruent in $e^{(3)}$. Once again, a small glitch in the motor decision appears in $t = 116$, caused by the experts outputs and the signal non-stationarity (a BARKING sound includes indeed some silence). The movement triggered at $t = 118$ is an error from the system since the object CRYING, baby should have been considered as congruent. The audio data perceived at this time is MALESPEECH, data from a never encountered audio class, thus enjoining the MFI module to trigger a head movement. From $t = 119$ and $t = 122$, the experts' data changed and became of class CRYING, dog, an audiovisual pair the MFI module never encountered before, consequently still promoting the focus on the object. However, at time $t = 123$, the correction of the MFI module has been applied and the "correct" audiovisual class CRYING, baby is now output by the module. The DW module, in response, analyses it and consider it as congruent in this environment, thus inhibiting the head movement. Finally, the new source SPEECH, male appears in the environment at $t = 150$ and the robot is focused on it. Two (apparently) erroneous movements to the resting position can be observed, at $t = 153$ and $t = 157$, due to the discontinuity of the sound signal: the audio experts

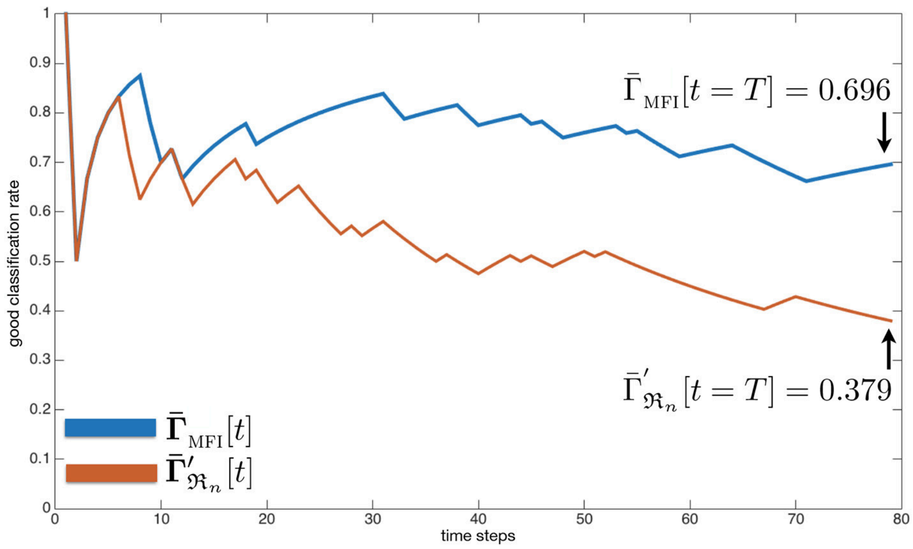


FIGURE 11 | Results of the audiovisual classification (including the inference by the MFI module) obtained by (blue) the HTM system, (red) the naive robot. The two numbers on the right correspond to the value at the end of the exploration.

did not detect any sound for these two frames (to give an idea: $\arg\max(\mathbf{P}^a[t = 157]) = 0.176$, whereas for the frame right before, at $t = 156$, five components out of thirteen are lying between $p^a = 0.403$ and $p^a = 1.00$). Going back to the resting position when an object stops emitting sound is part of the attempt of the overall HTM system to also inhibit the head movements in order to free the head for other potential purposes.

4.4. Evaluation 5: Fusion and Classification

After having performed numerous evaluation in simulated conditions (see **Table 2**), this experiment is focused on the evolution of the good audiovisual classification rate along the exploration of a real environment. For that purpose, an environment is set up with five different sources, as presented in **Table 3**. The three audio classes populating the environment have been selected because of their better experimental recognition rate in the architecture. At each time step, the estimated audiovisual classes provided by the overall HTM system is compared to the ground truth, and for each object. The resulting mean good estimation rate $\bar{\Gamma}_{\text{MFI}}[t]$, computed over all objects, is plotted against time in **Figure 11** (blue line). The same is done for the naive robot, with a mean good estimation rate $\bar{\Gamma}'_{\mathfrak{R}_n}[t]$ (red line in the same figure). As expected, the proposed HTM system shows the best audiovisual classification rate. Indeed, one can see in **Figure 11** that the red line tends to the rate $\bar{\Gamma}'_{\mathfrak{R}_n} = 37.9\%$ which is exactly the mean good classification rate of the involved KS. In the same conditions, the MFI module converges to $\bar{\Gamma}_{\text{MFI}} = 69.6\%$. In the very beginning of the experiment, both systems exhibit the same performances: the different smoothing involved in the various computations (of the KS outputs, in the motor decisions, etc.) together with silences in the sounds presented to the robot can explain this. But while the naive robot exhibits a constantly decreasing good estimation rate of the audiovisual classes, the MFI module remains relatively robust to the KS classification errors.

A direct consequence of these good performances of the HTM system can be observed in **Figure 12** which plots an histogram of all the audiovisual classes created by both systems (expressed in terms of number of frames). The HTM system is able to considerably narrow the possible audiovisual classes existing in the environment: from 22 by the naive robot, the HTM system narrows it down to only 5. However, one of the class created is erroneous: PIANO, `female` has been mistaken with PIANO, `male`, but only for a short period of time (two frames, i.e., 1 s). This point has already been discussed in section 3.3.4.

5. CONCLUSION

In this paper, a new system for the modulation of the exploratory behavior of a robot has been proposed. Based on the new notion of Congruence, it takes control of the head movements of a platform to put the robot attention toward audiovisual sources of interest. Additionally, it provides a robust description of the unknown environments explored all along the robot's life and following an unsupervised paradigm. This enriched representation consists, first, in the analysis of audiovisual objects through their relationship to the environments they are perceived in, and secondly, in how much the knowledge the system has about their actual audiovisual class is reliable and robust. Even in the case of classification errors by the audio or visual classifiers in the overall architecture, the system is

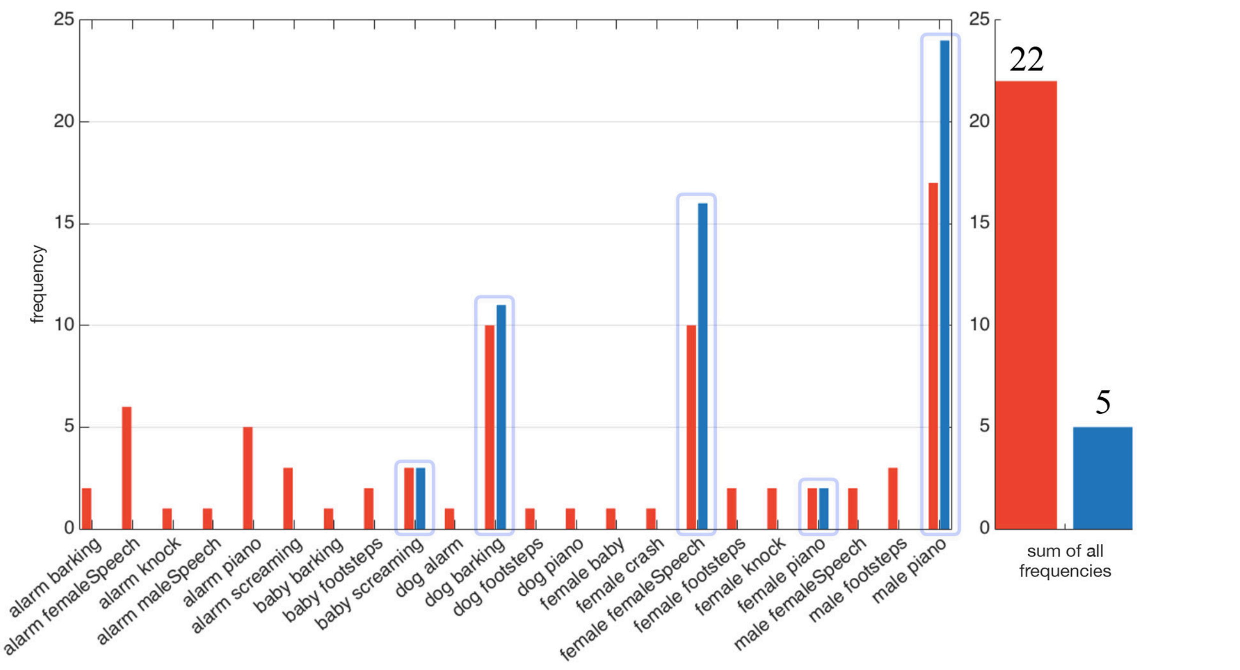


FIGURE 12 | Number and labels of the audiovisual classes created by (blue) the MFI module, and (red) the naive robot. **(Left)** Number of temporal frames (height of bars) during which the audiovisual classes have been categorized. (Light blue rectangles:) Audiovisual classes the two systems have in common. **(Right)** Total number of different audiovisual classes created.

then able to correctly infer the events' audiovisual classes by actively learning the interlink between the two modalities. All of this is achieved by the two constitutive modules of the HTM, namely the *Dynamic Weighting* module, and the *Multimodal Fusion & Inference* module. Each of them is able to trigger head movements that are used as an attentional reaction and as an active reaction to the need for additional data, respectively. Importantly, the extensive use of head movements is not limited to the sole benefit of the HTM system: audio localization algorithms such as (Nakashima and Mukai, 2005; Hornstein et al., 2006; Ma et al., 2017) relying also on head movements could be connected to the HTM as a top-down feedback unit, thus taking advantage from its motor commands to improve in parallel audio localization performances. The active self-supervised and online learning paradigm the MFI module relies upon, through the use of the *Multimodal Self- Organizing Map*, quickly provides the DW module with robust data while also offering inference abilities whenever a modality is missing (occlusion of the object, for instance). Whereas existing models provide audio-visual inference (Alameda-Pineda and Horaud, 2015) aiming at binding low-level cues of the audio and visual data streams, the MFI module relies only on a higher level of representation of data, a representation that could be used as a top-down feedback to potentially enhance low-level audiovisual fusion algorithm. Additionally, the choice of learning the cross-modal relationship between auditory and visual data in an exclusively unsupervised way can be debated as not being powerful enough (Senocak et al., 2018). However, the results obtained here show significant improvements in the quality of the audiovisual data provided to the DW module without any inclusion of human knowledge. The system performances have been evaluated in realistic simulated conditions, but also on a real robot endowed with binaural audition and vision capabilities. Importantly, the overall architecture of the system, i.e., the TWO!EARS software, is made available online as an open source software[2]. The same applies for the proposed HTM system, entirely included inside this architecture[3].

One of the main limitation of the current implementation is related to its high dependency to the localization experts. Indeed, the overall motor reactions are currently guided by each object azimuth localization, which have been shown precise enough to provide relevant results. Hopefully, binaural sound localization is a research topic by itself, and recent developments in the field show very robust algorithms, even in challenging acoustical conditions. Nevertheless, the robustness to localization errors could be enhanced by using tracking experts able to consolidate the sources position along time. For now, the HTM system is still being developed with the following improvements in mind. First, the definition of an object is currently limited to its audio and visual labels, while it could be enriched with additional information possibly coming from other modalities (emotions, audio pitch, forms and textures, etc.). Importantly, the proposed M-SOM has been designed to easily incorporate such additional parameters in the object definition: a subnetwork can be added for each of

[2]https://github.com/TWOEARS
[3]https://github.com/TWOEARS/audio-visual-integration

them together with their respective weights vectors. Concerning the Dynamic Weighting module, a significant improvement can be made by including the computation of a temporal habituation in order for the robot to not to be stuck in a deadlock kind of situation, as in **Figure 8** where, if the scenario goes on forever, the robot would be keeping turning its head toward the CRYING, `male`. Finally, the coupling of the HTM system with other cognitive experts in the current framework is still under investigation. So far, the current version of the TWO!EARS software does not include others high- level cognitive experts. Nevertheless, the entire HTM system has been conceived with the idea that the motor exploration can also be guided by cognitive elements other than the ones implemented in the system. For instance, a model as the one recently proposed by Lanillos et al. (2015) on attention driven by social interaction, could easily be linked to the HTM, both benefiting from each other: one congruent source could still be focused because of its social interest, whereas a socially non-interesting object could still be focused for its high incongruence.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

ACKNOWLEDGMENTS

This work is supported by the European FP7 TWO!EARS project, ICT-618075, www.twoears.eu.

REFERENCES

Alameda-Pineda, X., and Horaud, R. (2015). Vision-guided robot hearing. *Int. J. Robot. Res.* 34, 437–456. doi: 10.1177/0278364914548050

Baranes, A., and Oudeyer, P.-Y. (2009). R-IAC: robust intrinsically motivated active learning. *IEEE Trans. Auton. Ment. Dev.* 1, 155–169. doi: 10.1109/TAMD.2009.2037513

Baranes, A., and Oudeyer, P.-Y. (2010). "Intrinsically motivated goal exploration for active motor learning in robots: a case study," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on* (Taipei), 1766–1773. doi: 10.1109/IROS.2010.5651385

Bauer, J., and Wermter, S. (2013). "Self-organized neural learning of statistical inference from high-dimensional data," in *IJCAI* (Beijing), 1226–1232.

Berlyne, D. E. (1950). Novelty and curiosity as determinants of exploratory behavior. *Brit. J. Psychol.* 41, 68–80.

Berlyne, D. E. (1965). *Structure and Direction in Thinking*. New York, NY: John Wiley & Sons, Inc.

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. MIT Press.

Bustamante, G., Danès, P., Forgue, T., and Podlubne, A. (2016). "Towards information-based feedback control for binaural active localization," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Shanghai), 6325–6329.

Capdepuy, P., Polani, D., and Nehaniv, C. L. (2007). "Maximization of potential information flow as a universal utility for collective behaviour," in *IEEE Symposium on Artificial Life* (Honolulu, HI: IEEE), 207–213.

Carrillo, H., Dames, P., Kumar, V., and Castellanos, J. A. (2015). "Autonomous robotic exploration using occupancy grid maps and graph SLAM based on Shannon and Rényi entropy," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on* (Seattle, WA: IEEE), 487–494.

Chapelle, O., Weston, J., and Schölkopf, B. (2002). Cluster kernels for semi-supervised learning. *Adv. Neural Inform. Process. Syst. 15* 7:1.

Cohen-Lhyver, B. (2017). *Modulation de Mouvements de Tête pour l'Analyse Multimodale d'un Environnement Inconnu*. Ph.D. thesis, Université Pierre and Marie Curie.

Cohen-Lhyver, B., Argentieri, S., and Gas, B. (2015). "Modulating the auditory turn-to reflex on the basis of multimodal feedback loops: the dynamic weighting model," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)* (Buenos Aires).

Cohen-Lhyver, B., Argentieri, S., and Gas, B. (2016). "Multimodal fusion and inference using binaural audition and vision," in *International Congress on Acoustics*.

Corbetta, M., Patel, G., and Shulman, G. L. (2008). Review the reorienting system of the human brain: from environment to theory of mind. *Neuron* 58, 306–324. doi: 10.1016/j.neuron.2008.04.017

Corbetta, M., and Shulman, G. L. (2002). Control of goal-directed and stimulus-driven attention in the brain. *Nat. Rev. Neurosci.* 3, 201–215. doi: 10.1038/nrn755

Corneil, B. D., Munoz, D. P., Chapman, B. B., Admans, T., and Cushing, S. L. (2008). Neuromuscular consequences of reflexive covert orienting. *Nat. Neurosci.* 11:13. doi: 10.1038/nn2023

Cuperlier, N., Quoy, M., and Gaussier, P. (2007). Neurobiologically inspired mobile robot navigation and planning. *Front. Neurorobot.* 1:3. doi: 10.3389/neuro.12.003.2007

Downar, J., Crawley, A. P., Mikulis, D. J., and Davis, K. D. (2000). A multimodal cortical network for the detection of changes in the sensory environment. *Nat. Neurosci.* 3, 277–283. doi: 10.1038/72991

Driver, J., and Spence, C. (1998). Attention and crossmodal construction of space. *Trends Cogn. Sci.* 2, 254–262. doi: 10.1016/S1364-6613(98)01188-7

Droniou, A., Ivaldi, S., and Sigaud, O. (2015). Deep unsupervised network for multimodal perception, representation and classification. *Robot. Auton. Syst.* 71, 83–98. doi: 10.1016/j.robot.2014.11.005

Duangudom, V., and Anderson, D. V. (2007). "Using auditory saliency to understand complex auditory scenes," in *European Signal Processing Conference, 2017 (EUSIPCO)* (Poznan), number 15th.

Ferreira, J. F., and Dias, J. (2014). Attentional mechanisms for socially interactive robots–a survey. *IEEE Trans. Auton. Ment. Dev.* 6, 110–125. doi: 10.1109/TAMD.2014.2303072

Girard, B., Cuzin, V., Guillot, A., Gurney, K. N., and Prescott, T. J. (2002). "Comparing a brain-inspired robot action selection mechanism with 'Winner- Takes-All'," in *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*, Vol. 7 (Cambridge, MA: MIT Press), 75.

Gurney, K., Prescott, T. J., and Redgrave, P. (2001a). A computational model of action selection in the basal ganglia. I. A new functional anatomy. *Biol. Cybern.* 84, 401–410. doi: 10.1007/PL00007984

Gurney, K., Prescott, T. J., and Redgrave, P. (2001b). A computational model of action selection in the Basal Ganglia. II. analysis and simulation of behaviour. *Biol. Cybern.* 84, 411–423. doi: 10.1007/PL00007985

Henneberger, G., Brunsbach, B. J., and Klepsch, T. (1991). "Field oriented control of synchronous and asynchronous drives without mechanical sensors using a Kalman-Filter," in *European Conference on Power Electronics and Applications, 1991 (ECPEA)* (Firenze), Vol. 3, 664.

Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., et al. (2012). Gradient response maps for real-time detection of textureless objects. *IEEE Trans. Patt. Anal. Mach. Intell.* 34, 876–888. doi: 10.1109/TPAMI.2011.206

Hopfinger, J. B., Buonocore, M. H., and Mangun, G. R. (2000). The neural mechanisms of top-down attentional control. *Nat. Neurosci.* 3, 284–291. doi: 10.1038/72999

Hornstein, J., Lopes, M., Santos-Victor, J., and Lacerda, F. (2006). "Sound localization for humanoid robots-building audio-motor maps based on the hrtf," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* (Beijing: IEEE) , 1170–1176.

Huang, G. B., Member, S., Zhu, Q. Y., and Siew, C. K. (2006). Real-time learning capability of neural networks. *IEEE Trans. Neural Netw.* 17, 863–878. doi: 10.1109/TNN.2006.875974

Huang, X., and Weng, J. (2002). "Novelty and reinforcement learning in the value system of developmental robots," in *Proceedings of the Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems* (Edinburgh: Lund University Cognitive Studies), 47–55.

Huang, X., and Weng, J. (2004). "Motivational system for human-robot interaction," in *International Workshop on Computer Vision in Human-Computer Interaction* (Prague: Springer), 17–27.

Hunt, J. (1965). "Intrinsic motivation and its role in psychological development," in *Nebraska Symposium on Motivation*, Vol. 13 (University of Nebraska Press), 189–282.

Indovina, I., and Macaluso, E. (2007). Dissociation of stimulus relevance and saliency factors during shifts of visuospatial attention. *Cereb. Cortex* 17, 1701–1711. doi: 10.1093/cercor/bhl081

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Patt. Anal. Mach. Intell.* 20, 1254–1259. doi: 10.1109/34.730558

James, W. (1890). *The Principles of Psychology*. Read Books Ltd.

Kayser, C., Petkov, C. I., Lippert, M., and Logothetis, N. K. (2005). Mechanisms for allocating auditory attention: an auditory saliency map. *Curr. Biol.* 15, 1943–1947. doi: 10.1016/j.cub.2005.09.040

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biol. Cybern.* 43, 59–69. doi: 10.1007/BF00337288

Kohonen, T. (1990). The self-organizing map. *Proc. IEEE* 78, 1464–1480. doi: 10.1109/5.58325

Kohonen, T. (2013). Essentials of the self-organizing map. *Neural Netw.* 37, 52–65. doi: 10.1016/j.neunet.2012.09.018

Lanillos, P., Ferreira, J. F., and Dias, J. (2015). "Designing an artificial attention system for social robots," in *Institute of Electrical and Electronics Engineers (IEEE)/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Hamburg). doi: 10.1109/IROS.2015.7353967

Le Meur, O., Le Callet, P., Barba, D., and Thoreau, D. (2006). A coherent computational approach to model bottom-up visual attention. *Trans. Patt. Anal. Mach. Intell.* 28, 802–817. doi: 10.1109/TPAMI.2006.86

Le Meur, O., and Liu, Z. (2015). Saccadic model of eye movements for free-viewing condition. *Vis. Res.* 116, 152–164. doi: 10.1016/j.visres.2014.12.026

Ma, N., Brown, G. J., and May, T. (2015). "Exploiting deep neural networks and head movements for binaural localisation of multiple speakers in reverberant conditions," in *Interspeech* (Dresden).

Ma, N., May, T., and Brown, G. J. (2017). Exploiting deep neural networks and head movements for robust binaural localization of multiple sources in reverberant environments. *IEEE/ACM Trans. Audio Speech Lang. Process. TASLP)* 25, 2444–2453. doi: 10.1109/TASLP.2017.2750760

Makarenko, A. A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F. (2002). "An experiment in integrated exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002* (Lausanne), 534–539. doi: 10.1109/IRDS.2002.1041445

May, T., van de Par, S., and Kohlrausch, A. (2011). A probabilistic model for robust localization based on a binaural auditory front-end. *IEEE Trans. Audio Speech Lang. Process.* 19, 1–13. doi: 10.1109/TASL.2010.2042128

Meyer, J.-A., Guillot, A., Girard, B., Khamassi, M., Pirim, P., and Berthoz, A. (2005). The psikharpax project: towards building an artificial rat. *Robot. Auton. Syst.* 50, 211–223. doi: 10.1016/j.robot.2004.09.018

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). "FastSLAM: a factored solution to the simultaneous localization and mapping problem," in *Proceedings of 8th National Conference on Artificial Intelligence/14th Conference on Innovative Applications of Artificial Intelligence*, Vol. 68 (Edmonton, AB), 593–598.

Näätänen, R., Gaillard, A. W., and Mäntysalo, S. (1978). Early selective-attention effect on evoked potential reinterpreted. *Acta Psychol.* 42, 313–329. doi: 10.1016/0001-6918(78)90006-9

Nakashima, H., and Mukai, T. (2005). "3d sound source localization system based on learning of binaural hearing," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, Vol. 4 (London, UK: IEEE), 3534–3539.

Nothdurft, H.-C. (2006). Salience and target selection in visual search. *Vis. Cogn.* 14, 514–542. doi: 10.1080/13506280500194162

O'Keefe, J., and Nadel, L. (1978). *The Hippocampus as a Cognitive Map*. Number 9. Oxford: Clarendon Press.

Oliva, A., Torralba, A., Castelhano, M. S., and Henderson, J. M. (2003). "Top-down control of visual attention in object detection," in *2003 International Conference on Image Processing, 2003,* Vol. 1 (ICIP 2003) (Barcelona), 1–253.

Oudeyer, P.-Y., and Kaplan, F. (2008). "How can we define intrinsic motivation?" in *Proceedings of the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Lund University Cognitive Studies, Lund: LUCS, Brighton*. Brighton; Lund: Lund University Cognitive Studies.

Oudeyer, P. Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evolut. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Papliński, A. P., and Gustafsson, L. (2005). "Multimodal feedforward self-organizing maps," in *International Conference on Computational and Information Science* (Xi'an: Springer), 81–88.

Petersen, S., and Posner, M. (2012). The attention system of the human brain: 20 years after. *Annu. Rev. Neurosci.* 21, 73–89. doi: 10.1146/annurev-neuro-062111-150525

Redgrave, P., Prescott, T. J., and Gurney, K. (1999). The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience* 89, 1009–1023. doi: 10.1016/S0306-4522(98)00319-4

Roy, N., McCallum, A., and Com, M. W. (2001). "Toward optimal active learning through Monte Carlo estimation of error reduction," in *International Conference on Machine Learning* (Williamstown, VIC).

Ruesch, J., Lopes, M., Bernardino, A., Hörnstein, J., Santos-Victor, J., and Pfeifer, R. (2008). "Multimodal saliency-based bottom-up attention a framework for the humanoid robot iCub," in *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008* (Pasadena, CA), 962–967.

Rushworth, M. F., Noonan, M. P., Boorman, E. D., Walton, M. E., and Behrens, T. E. (2011). Frontal cortex and reward-guided learning and decision-making. *Neuron* 70, 1054–1069. doi: 10.1016/j.neuron.2011.05.014

Ryan, R. M., and Deci, E. L. (2000). Intrinsic and extrinsic motivations: classic definitions and new directions. *Contemp. Educ. Psychol.* 25, 54–67. doi: 10.1006/ceps.1999.1020

Schillaci, G., Hafner, V. V., and Lara, B. (2014). "Online learning of visuo-motor coordination in a humanoid robot. a biologically inspired model," in *Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on* (Genoa: IEEE), 130–136.

Schmidhuber, J. (1991). "Curious model-building control systems," in *Neural Networks, 1991. 1991 IEEE International Joint Conference on* (Singapore), 1458–1463.

Schymura, C., Walther, T., Kolossa, D., Brown, G. J., and Ma, N. (2014). Binaural sound source localisation using a Bayesian-network-based blackboard system and hypothesis-driven feedback. *Fourm Acusticum*. doi: 10.13140/2.1.4026.4966

Senocak, A., Oh, T.-H., Kim, J., Yang, M.-H., and Kweon, I. S. (2018). "Learning to localize sound source in visual scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT), 4358–4366.

Smith, R., Self, M., and Cheeseman, P. (1987). "A stochastic map for uncertain spatial relationships," in *4th International Symposium on Robotic Research*, 467–474.

Thompson, G. C., and Masterton, R. B. (1978). Brain stem auditory pathways involved in reflexive head orientation to sound. *J. Neurophysiol.* 41, 1183–1202. doi: 10.1152/jn.1978.41.5.1183

Treisman, A. M., and Gelade, G. (1980). A feature-integration theory of attention. *Cogn. Psychol.* 12, 97–136. doi: 10.1016/0010-0285(80)90005-5

Tsiami, A., Katsamanis, A., Maragos, P., and Vatakis, A. (2016). "Towards a behaviorally-validated computational audiovisual saliency model," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (Shanghai), 2847–2851.

Two!Ears (2016a). *Report on Evaluation of the Two!Ears Expert System*. Technical Report June. Two!Ears Consortium.

Two!Ears (2016b). *The Two!Ears Project*.

Walter, W. G. (1951). A machine that Learns. *Sci. Amer*. 185, 60–63. doi: 10.1038/scientificamerican0851-60

Walther, D., Rutishauser, U., Koch, C., and Perona, P. (2005). Selective visual attention enables learning and recognition of multiple objects in cluttered scenes. *Comput. Vis. Image Underst*. 100, 41–63. doi: 10.1016/j.cviu.2004.09.004

Walther, T., and Cohen-Lhyver, B. (2014). "Multimodal feedback in auditory-based active scene exploration," in *Forum Acusticum* (Krakow).

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). "Learning with local and global consistency," in *Advances in Neural Information Processing Systems 17, 2004. Proceedings of Neural Information Processing Systems*, Vol. 1.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

PERSPECTIVE
published: 25 September 2018
doi: 10.3389/fnbot.2018.00059

Open-Ended Learning: A Conceptual Framework Based on Representational Redescription

Stephane Doncieux[1]*, *David Filliat*[2], *Natalia Díaz-Rodríguez*[2], *Timothy Hospedales*[3], *Richard Duro*[4], *Alexandre Coninx*[1], *Diederik M. Roijers*[5], *Benoît Girard*[1], *Nicolas Perrin*[1] *and Olivier Sigaud*[1]

[1] Sorbonne Université, CNRS, ISIR, Paris, France, [2] U2IS, INRIA Flowers, ENSTA ParisTech, Palaiseau, France, [3] Institute of Perception, Action and Behaviour, University of Edinburgh, Edinburgh, United Kingdom, [4] GII, Universidade da Coruña, A Coruña, Spain, [5] Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, Netherlands

Reinforcement learning (RL) aims at building a policy that maximizes a task-related reward within a given domain. When the domain is known, i.e., when its states, actions and reward are defined, Markov Decision Processes (MDPs) provide a convenient theoretical framework to formalize RL. But in an open-ended learning process, an agent or robot must solve an unbounded sequence of tasks that are not known in advance and the corresponding MDPs cannot be built at design time. This defines the main challenges of open-ended learning: how can the agent learn how to behave appropriately when the adequate states, actions and rewards representations are not given? In this paper, we propose a conceptual framework to address this question. We assume an agent endowed with low-level perception and action capabilities. This agent receives an external reward when it faces a task. It must discover the state and action representations that will let it cast the tasks as MDPs in order to solve them by RL. The relevance of the action or state representation is critical for the agent to learn efficiently. Considering that the agent starts with a low level, task-agnostic state and action spaces based on its low-level perception and action capabilities, we describe open-ended learning as the challenge of building the adequate representation of states and actions, i.e., of redescribing available representations. We suggest an iterative approach to this problem based on several successive Representational Redescription processes, and highlight the corresponding challenges in which intrinsic motivations play a key role.

Keywords: developmental robotics, reinforcement learning, state representation learning, representational redescription, actions and goals, skills

Edited by:
Andrew Barto,
University of Massachusetts Amherst, United States

Reviewed by:
Robert J. Lowe,
University of Gothenburg, Sweden
Georg Martius,
Max-Planck-Institut für Intelligente Systeme, Germany
Eiji Uchibe,
Advanced Telecommunications Research Institute International (ATR), Japan

****Correspondence:***
Stephane Doncieux
stephane.doncieux@sorbonne-universite.fr

Received: *28 April 2018*
Accepted: *28 August 2018*
Published: *25 September 2018*

Citation:
Doncieux S, Filliat D, Díaz-Rodríguez N, Hospedales T, Duro R, Coninx A, Roijers DM, Girard B, Perrin N and Sigaud O (2018) Open-Ended Learning: A Conceptual Framework Based on Representational Redescription. Front. Neurorobot. 12:59. doi: 10.3389/fnbot.2018.00059

1. INTRODUCTION

Robots need world representations in terms of objects, actions, plans, etc. Currently such representations are carefully designed and adapted to the robot's task (Kober et al., 2013). But a general purpose robot capable of solving an unbounded number of tasks cannot rely on representations hardwired at design time, because each may require a different representation. To achieve the vision of a robot that can solve an open-ended series of tasks in an increasingly efficient way, we consider an alternative paradigm: that the robot should discover the appropriate representations required to autonomously learn each task.

Representational redescription is the ability to discover new representations based on existing ones. It is a key ability of human intelligence (Karmiloff-Smith, 1995) that remains a challenge in robotics (Oudeyer, 2015). In this paper, we propose a unifying conceptual framework for addressing it. We assume an agent endowed with low-level perception and action capabilities which receives external rewards when it faces a task. We also assume the agent is endowed with reinforcement learning (RL) capabilities efficient enough to let it learn to solve a task when cast as a Markov Decision Process (MDP). From these assumptions, the main challenge in our framework is determining how an agent can discover the state and action representations that let it cast tasks as MDPs, before solving them by RL (Zimmer and Doncieux, 2018).

In MDPs, states and actions are primitive components considered given, and they are generally defined by the human designer having a particular task and domain in mind (see **Figure 1**). To make a step toward open-ended learning, we propose a conceptual framework for representational redescription processes based on a formal definition of states and actions. Then we highlight the challenges it raises, notably in terms of intrinsic motivations.

2. THE REPRESENTATIONAL REDESCRIPTION APPROACH

Our Representational Redescription approach is depicted in **Figure 2**. We consider an agent endowed with low-level perception and action capabilities, and which faces an open-ended sequence of tasks. The agent receives some external rewards from these tasks. The problem for this agent is to determine how to use this reward to learn the corresponding task. In an MDP, an RL algorithm explores the possible outcomes of an action when executed in a particular state. As pointed out by Kober et al. (2013), there is a need to appropriately define the state and action spaces for an efficient learning process. To do so, the possible alternatives are either to rely on a single generic state and action space or to build them on-the-fly when required. In this work, we do the latter and make the following assumptions:

ASSUMPTION 1. *A* single *state and action space cannot efficiently represent all the decision processes required to solve the tasks an open-ended learning system will be confronted to. To solve the task k defined through a reward value $r_k(t)$, the agent needs to build an MDP M_k.*

ASSUMPTION 2. *An open-ended learning process needs to build these MDPs on-the-fly.*

ASSUMPTION 3. *The agent is endowed with some RL algorithms to allow it to learn to solve the task, once the underlying MDP has been fully defined.*

3. CONCEPTUAL FRAMEWORK AND BASIC DEFINITIONS

3.1. Markov Decision Processes

Decisions in robotics can be modeled with MDPs using $< S_k, A_k, p_k, R_k >$, where k is a task identifier[1], S_k is the state space, A_k is the action space, $p_k : S_k \times A_k \times S_k \rightarrow \mathbb{R}$ is a transition function, where $p_k(s_t, a_t, s_{t+1})$ gives the probability to reach s_{t+1} from s_t after having applied action a_t and $R_k : S_k \rightarrow \mathbb{R}$ is the reward function. A policy $\pi_k : S_k \rightarrow A_k$ is a process that determines which action to apply in any state.

In the proposed framework, the observed reward $r_k(t)$ is distinguished from the reward function of the MDP $R_k(t)$. The agent may not know to what state the observed reward $r_k(t)$ can be associated. It is actually part of the proposed open-ended learning framework to *interpret* observed rewards and associate them to states in order to build the reward function $R_k(t)$ required to learn how to maximize them.

The notations used here have been intentionally kept as simple as possible. This framework can be easily extended to more complex cases, including semi-MDPs, stochastic policies or other definitions of the reward function.

3.2. States

DEFINITION 1. *A* state *is a description of a robot context that respects the constraints of its decision process.*

Following (Lesort et al., 2018), a good state representation should be (1) Markovian (i.e., the current state summarizes all the necessary information to choose an action), (2) able to represent the robot context well enough for policy improvement, (3) able to generalize the learned value-function to unseen states with similar features, and (4), low dimensional for efficient estimation (Böhmer et al., 2015). State representation learning approaches learn low dimensional representations without direct supervision, i.e., exploiting sequences of observations, actions, rewards and generic learning objectives (Lesort et al., 2018).

To bootstrap the open-ended learning process, we define S_0 as the state space containing the set of possible sensorimotor values. This first state space may not be low dimensional, Markovian, or structured enough for efficient exploration, thus motivating the search for better adapted state spaces.

3.3. Reward Functions and Goals

A reward function may contain different kinds of information: an indication of success in fulfilling a Human user defined task, or in reaching an autonomously defined action goal (see next section). It may also contain guidance to help reach the goal (reward shaping).

Besides reward functions defined in $\mathbb{R}$, the proposed framework requires, for the description of actions, the definition of boolean reward functions that will be called *goal functions*:

[1] A single state and action space can be used for several tasks and a single task could be associated to multiple representations, but we use this notation to highlight the dependency between tasks and MDPs that is central to this framework.

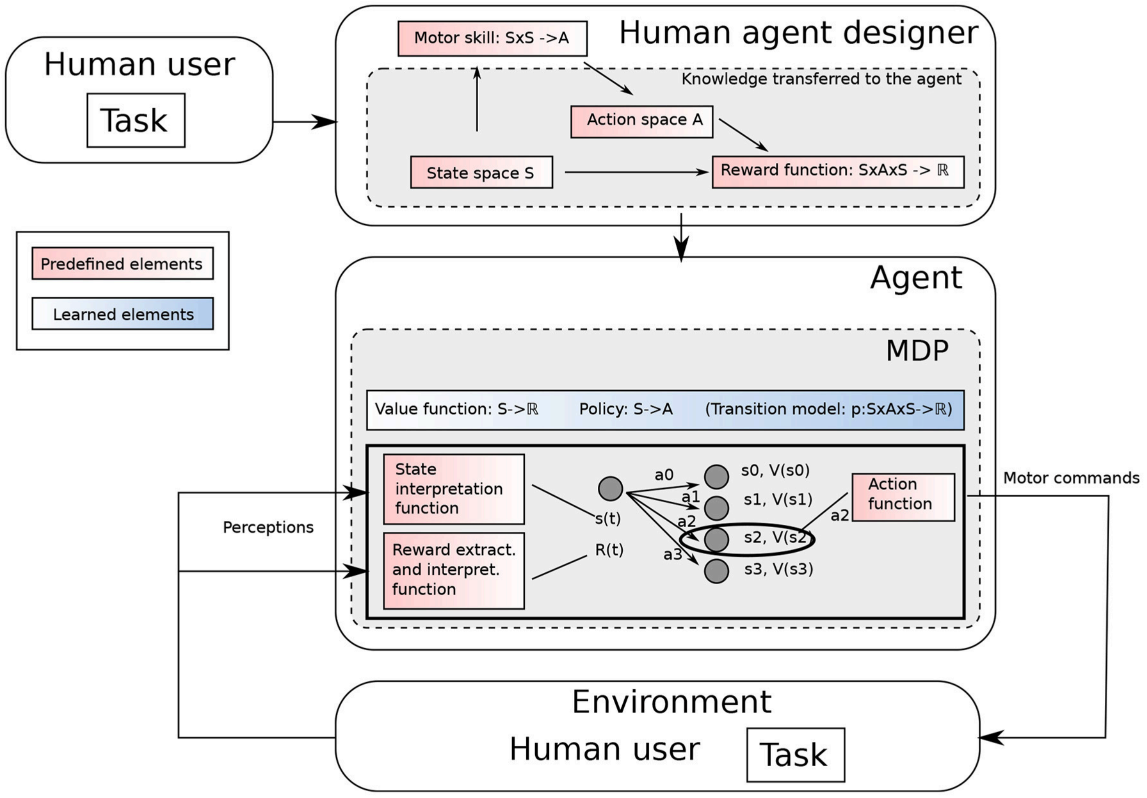


FIGURE 1 | A typical MDP. The agent designer having a task in mind designs the MDP accordingly.

DEFINITION 2. *A* goal function, *denoted $\hat{R}$, does not contain any shaping term and tells whether the goal associated to this reward function is achieved or not.*

A goal function is a specific reward function aimed at defining the notion of success or failure required for action definition. The task to solve does not need to be described with such a boolean function.

DEFINITION 3. Goal states *are states s for which $\hat{R}(s) = True$.*

3.4. Actions

In the proposed framework, actions are not systematically predefined, but can be built on-the-fly. The design of the corresponding algorithms requires to define what an action actually is. The proposed definition relies on the notion of goal function to add a purpose to a policy. Actions are framed within different abstraction levels depending on the granularity of the policy, as in the options framework (Sutton et al., 1999). Actions are one of the main components of an MDP. An MDP M_k needs an action space A_k. A_k is an action space defined at an abstraction level k. It relies on policies of level $k-1$, defined in an MDP $k-1$. They can be used to build policies at the level k that can, in turn, be used to build new actions for another MDP at the level $k+1$.

DEFINITION 4. *Actions $a \in A_k$ are the primitives of MDP M_k. An* action *a is a policy π relying on actions available at a lower level and built to reach a goal state associated to a goal function $\hat{R}$. The action succeeds if the trajectory of the robot controlled by this policy converges to a goal state of $\hat{R}$; otherwise, it fails. An action is then fully defined by the triplet: $\{\pi, \hat{R}, t_{max}\}$ where t_{max} is the maximum amount of time after which the action is considered failed if no goal state is reached.*

If the level on top of which an MDP M_k is built is, itself, an MDP, actions $a \in A_k$ can be considered as macro-actions or options.

The goal state of an action is frequently defined relative to a particular initial state s_{init}, where s_{init} is the state of the agent when the action is triggered, e.g., "Turning 90*deg*" or "moving forward 1*m*."

The definition of an action is hierarchically recurrent: an action a_k relies on a policy π that also relies on a set of lower level actions $\{a_l, l < k\}$. To stop the recurrence, a specific set of actions A_0 is defined, that corresponds to the lowest level accessible by the robot, i.e., motor commands. They are also actions, as motor commands always have a goal (reaching a particular velocity or position, for instance) that a low-level control process aims at reaching and eventually staying at. As suggested by Harutyunyan (2018, Chapter 5), we assert that it may not be necessary, or even desirable, to have the same time-scale and discounting for lower and higher level actions.

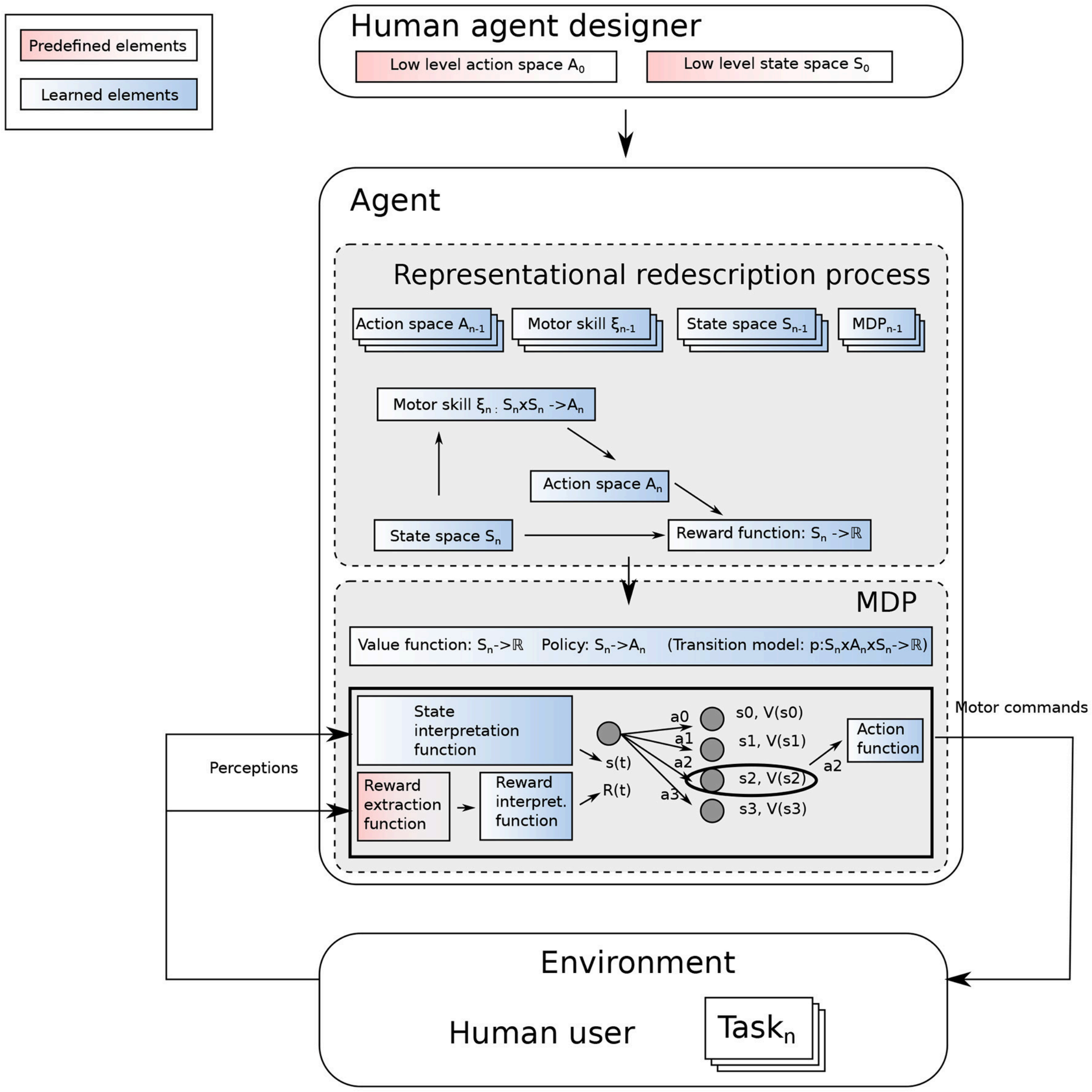


FIGURE 2 | Overview of an open-ended learning process. The agent designer does not know the different tasks the agent will be facing, but designs the agent to let it build the MDP to interpret a reward in its environment and find out how to maximize it.

3.5. Representational Redescription

In the proposed framework, open-ended learning needs to build MDPs on-the-fly, including the state and action spaces. Considering that the process starts from initial state and action spaces (S_0, A_0), this particular feature is captured by the concept of *representational redescription*.

DEFINITION 5. *A* representational redescription process *is a process that builds the state and action spaces enabling the definition of an MDP able to (1) solve a given task defined by observed reward values (2) in a particular domain and (3) with a particular decision or learning process. To this end, it relies on the representations of states and actions that have been previously acquired and can thus be described as a process transforming existing representations into new ones that are more fitted to the context.*

3.6. Motor Skills: Controlling Transitions Between States

In an MDP, the set of provided actions is built to allow the robot to move in the state space. If a state space is built on-the-fly, the agent should be able to control it and move from one state to another. With the proposed definitions, the open-ended learning process needs to build actions to reach each part of the state space. The notion of motor skill is defined to capture this process.

DEFINITION 6. *A* motor skill *is an action generator:* $\xi_k : S^{(i)} \times S^{(g)} \rightarrow A_k$, *where* $S^{(i)}, S^{(g)} \subset S_k^2$. *It is an inverse model defined in* *a particular action space A_k to reach a target state from an initial state.*

$\xi(s_i, s_g)$ is an action that, starting from $s_i \in S^{(i)}$, reaches $s_g \in S^{(g)}$ with the highest possible probability. The state s_g is the goal state of the corresponding action, and the corresponding reward function is intrinsic (see section 3.8).

3.7. Open-Ended Learning

On the basis of the proposed definitions, we can define an open-ended learning process as follows:

DEFINITION 7. *An* open-ended learning process *builds the MDPs required to solve the tasks that the agent faces. Task k is defined through an observed reward value $r_k(t)$. Starting from an initial state space S_0, an initial action space A_0 and a decision or learning process $\mathcal{P}$, the open-ended learning process aims at building (1) state spaces, (2) action spaces and (3) motor skills to control the state with appropriate actions. State and action spaces need to fulfil the following features:*

1. *The state space should help interpret the reward occurences, i.e., learn R_k to model the observed r_k;*
2. *The action space should allow control of the state space through dedicated motor skills;*
3. *The state and action spaces should make the agent's state trajectory as predictable as possible;*
4. *From the state and action spaces, $\mathcal{P}$ should be able to converge to a policy maximizing r_k.*

3.8. Intrinsic Motivations

Task-based rewards are not enough to drive a representational redescription process. There is a need for other drives that push the agent to explore and create new knowledge. This is the role of intrinsic motivations (Oudeyer and Kaplan, 2009; Baldassarre and Mirolli, 2013). In the context of open-ended learning through representational redescription, we propose the following definition:

DEFINITION 8. *An* intrinsic motivation *is a drive that complements drives associated with external task-based rewards to build appropriate state and actions spaces as well as motor skills.*

Intrinsic motivations play a critical role at different stages of the proposed representational redescription process, for instance:

- To organize learning and select in what order to learn skills and build state spaces;
- To acquire relevant data for state representation learning (before building an appropriate MDP);
- To build the skills required to control the state space (focusing learning on areas that are within reach and ignoring the rest).

4. CHALLENGES

This section recasts the challenges of open-ended learning with the proposed conceptual framework.

CHALLENGE 1. *Interpreting observed reward: Building an appropriate state space to interpret an externally provided reward, i.e., build a state space S_k that allows easy modeling of the observed reward value r_k.*

CHALLENGE 2. *Skill acquisition: Controlling the displacements in an acquired state space S_k by building the appropriate action space A_k and skill $\xi_k : S^{(i)} \times S^{(g)} \rightarrow A_k$, where $S^{(i)}, S^{(g)} \subset S_k^2$, to give the agent the ability to move from one state to another as accurately as possible.*

To address Challenge 1, state representations can be learned from known actions (Jonschkowski and Brock, 2015) and, likewise, to address Challenge 2, actions can be learned when the state space is known (Rolf et al., 2010; Forestier et al., 2017).

CHALLENGE 3. *Simultaneously learning state space, action space, and policy: The state and action spaces are interdependent with each other and with the policy. For open-ended learning, all need to be learned jointly to solve a task, and doing so tractably is a challenge.*

CHALLENGE 4. *Dealing with sparse reward: available state and action spaces may not allow to easily obtain reward $r_k(t)$ associated to Task k. This is particularly true at the beginning of the process, when starting from (S_0, A_0): this is the bootstrap problem. The challenge is to design an exploration process that converges toward reward observations in a limited time.*

A possibility to address the bootstrap problem is to rely on a motor babbling approach (Baranes and Oudeyer, 2010; Rolf et al., 2010). Another possibility would be to rely on a direct policy search including an intrinsic motivation toward behavior diversity and followed by a process to extract adapted representations from it (Zimmer and Doncieux, 2018).

The next challenges are related to the unsupervised acquisition of a hierarchy of adapted representations.

CHALLENGE 5. *Detecting task change: In the case where tasks are not explicitly indicated to the robot, detecting task change from k to $k+1$ on the basis of observed rewards r.*

The efficiency of a learning system is influenced by the order of the tasks it is facing (Bengio et al., 2009).

CHALLENGE 6. *Ordering knowledge acquisition and task resolution: An open-ended learning system needs to be able to select what to focus on and when. Does it keep learning representations for task k (even if r_k has momentarily disappeared), or does it focus on a new task $k+1$?*

CHALLENGE 7. *Identifying the available knowledge relevant to build the new representations MDP_k: as the set of available MDPs grows, it becomes a challenge to figure out what knowledge can help to build a new and adapted representation, i.e., $\{MDP_l, \pi_l\}_{l \leq k} = \{< S_l, A_l, p_l, R_l >, \pi_l\}_{l \leq k}$.*

CHALLENGE 8. *Using transfer learning for speeding up state and action spaces learning along time: as the number of tasks and domains the agent can deal with grows, it becomes interesting when facing a task $k+1$ to reuse the knowledge available to avoid learning MDP_{k+1} and π_{k+1} from scratch.*

5. DISCUSSION

In contrast to many works in multitask learning (Zhao et al., 2017; Riedmiller et al., 2018), we assume here that each task should be solved with its own state and action representation, and learning these representations is a central challenge. We adopt a hierarchical perspective based on representational redescription which differs from the hierarchical RL perspective (Barto and Mahadevan, 2003) from the fact that we do not assume that the lowest level is necessarily described as an MDP and we assume that each intermediate level may come with its own representation.

The proposed framework is related to end-to-end approaches to reinforcement learning (Lillicrap et al., 2015; Levine et al., 2016), but instead of black box approaches, it emphasizes knowledge reuse through the explicit extraction of relevant representations.

Open-ended learning is expected to occur in a lifelong learning scenario in which the agent will be confronted with multiple challenges to build the knowledge required to solve the tasks it will face. It will not be systematically engaged in a task resolution problem and will thus have to perform choices that cannot be guided by a reward. Intrinsic motivations are thus a critical component of the proposed open-ended learning system. They will fill in multiple needs of such a system: (1) a drive for action and state space acquisition (Péré et al., 2018), (2) a selection of what to focus on (Oudeyer et al., 2007) and (3) a bootstrap of the process in the case of sparse reward (Mouret and Doncieux, 2012).

AUTHOR CONTRIBUTIONS

This article is the result of a joint work within the DREAM project. Each author has participated to the discussions that have lead to the proposed formalism. SD has coordinated the discussions and the writing.

ACKNOWLEDGMENTS

This work has been supported by the FET project DREAM[2], that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 640891.

[2]http://www.robotsthatdream.eu/

REFERENCES

Baldassarre, G., and Mirolli, M. (eds.). (2013). *Intrinsically Motivated Learning in Natural and Artificial Systems*. Berlin; Heidelberg: Springer-Verlag.

Baranes, A., and Oudeyer, P.-Y. (2010). "Intrinsically motivated goal exploration for active motor learning in robots: a case study," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (*IROS 2010*) (Taipei).

Barto, A., and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Syst.* 13, 41–77. doi: 10.1023/A:1022140919877

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, QC: ACM), 41–48.

Böhmer, W., Springenberg, J. T., Boedecker, J., Riedmiller, M., and Obermayer, K. (2015). Autonomous learning of state representations for control: an emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. *KI-Künstliche Intell.* 29, 353–362. doi: 10.1007/s13218-015-0356-1

Forestier, S., Mollard, Y., and Oudeyer, P.-Y. (2017). Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*.

Harutyunyan, A. (2018). *Beyond Single-Step Temporal Difference Learning*. Ph.D. thesis, Vrije Universiteit Brussel (VUB).

Jonschkowski, R., and Brock, O. (2015). Learning state representations with robotic priors. *Auton. Robots* 39, 407–428. doi: 10.1007/s10514-015-9459-7

Karmiloff-Smith, A. (1995). *Beyond Modularity: A Developmental Perspective on Cognitive Science*. Cambridge, MA: MIT Press.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: a survey. *Int. J. Robot. Res.* 32, 1238–1274. doi: 10.1177/0278364913495721

Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., and Filliat, D. (2018). State representation learning for control: an overview. *arXiv preprint arXiv:1802.04181*.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* 17, 1334–1373. Available online at: http://www.jmlr.org/papers/v17/15-522.html

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mouret, J.-B., and Doncieux, S. (2012). Encouraging behavioral diversity in evolutionary robotics: an empirical study. *Evol. Comput.* 20, 91–133. doi: 10.1162/EVCO_a_00048

Oudeyer, P.-Y. (ed.). (2015). *IEEE CIS Newsletter on Cognitive and Developmental Systems. Representational Redescription: The Next Challenge?* Vol. 12)(IEEE). Available online at: https://openlab-flowers.inria.fr/t/ieee-cis-newsletter-on-cognitive-and-developmental-systems/129

Oudeyer, P.-Y., and Kaplan, F. (2009). What is intrinsic motivation? A typology of computational approaches. *Front. Neurorobot.* 1:6. doi: 10.3389/neuro.12.006.2007

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evol. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Péré, A., Forestier, S., Sigaud, O., and Oudeyer, P.-Y. (2018). Unsupervised learning of goal spaces for intrinsically motivated goal exploration. *arXiv preprint arXiv:1803.00781*.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Van de Wiele, T., et al. (2018). Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*.

Rolf, M., Steil, J. J., and Gienger, M. (2010). Goal babbling permits direct learning of inverse kinematics. *IEEE Trans. Auton. Mental Dev.* 2, 216–229. doi: 10.1109/TAMD.2010.2062511

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.* 112, 181–211. doi: 10.1016/S0004-3702(99)00052-1

Zhao, C., Hospedales, T. M., Stulp, F., and Sigaud, O. (2017). "Tensor based knowledge transfer across skill categories for robot control," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (AAAI Press), 3462–3468.

Zimmer, M., and Doncieux, S. (2018). Bootstrapping *q*-learning for robotics from neuro-evolution results. *IEEE Trans. Cogn. Dev. Syst.* 10, 102–119. doi: 10.1109/TCDS.2016.2628817

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 27 September 2018
doi: 10.3389/fnbot.2018.00061

Cooperative and Competitive Reinforcement and Imitation Learning for a Mixture of Heterogeneous Learning Modules

*Eiji Uchibe**

Advanced Telecommunications Research Computational Neuroscience Laboratories, Department of Brain Robot Interface, Kyoto, Japan

This paper proposes Cooperative and competitive Reinforcement And Imitation Learning (CRAIL) for selecting an appropriate policy from a set of multiple heterogeneous modules and training all of them in parallel. Each learning module has its own network architecture and improves the policy based on an off-policy reinforcement learning algorithm and behavior cloning from samples collected by a behavior policy that is constructed by a combination of all the policies. Since the mixing weights are determined by the performance of the module, a better policy is automatically selected based on the learning progress. Experimental results on a benchmark control task show that CRAIL successfully achieves fast learning by allowing modules with complicated network structures to exploit task-relevant samples for training.

Keywords: reinforcement learning, imitation learning, modular architecture, parallel learning, entropy-regularization, multiple importance sampling

OPEN ACCESS

Edited by:
Andrew Barto,
University of Massachusetts Amherst, United States

Reviewed by:
Gerhard Neumann,
Technische Universität Darmstadt, Germany
Luca Citi,
University of Essex, United Kingdom

***Correspondence:**
Eiji Uchibe
uchibe@atr.jp

Received: *31 May 2018*
Accepted: *06 September 2018*
Published: *27 September 2018*

Citation:
Uchibe E (2018) Cooperative and Competitive Reinforcement and Imitation Learning for a Mixture of Heterogeneous Learning Modules. Front. Neurorobot. 12:61. doi: 10.3389/fnbot.2018.00061

1. INTRODUCTION

Reinforcement Learning (RL) (Sutton and Barto, 1998; Kober et al., 2013) is an attractive learning framework with a wide range of possible application areas. A learning agent attempts to find a policy that maximizes its total amount of reward received during interaction with its environment. Recently, such nonlinear function approximators as artificial neural networks are being used to approximate a policy with the help of deep learning. Deep Reinforcement Learning (DRL), which integrates both deep learning and reinforcement learning, has achieved several remarkable successes in decision-making tasks, such as playing video games (Mnih et al., 2015) and the board game Go (Silver et al., 2016, 2017).

However, DRL's performance critically depends on its architectures, learning algorithms, and meta-parameters (Henderson et al., 2018). On one hand, a shallow Neural Network (NN) with fewer connection weights usually learns faster, but its performance may be limited. A deep and/or wide NN with many network weights can represent any complex policy, but it usually needs a huge amount of experiences to find an appropriate one. Since the motivation to use NNs is to represent complicated nonlinear mapping from state to action, it is reasonable to select a deep and wide NN as a function approximator. However, training data must be gathered by the learning agent for reinforcement learning as opposed to the standard settings of the classification problems of deep learning. Since a complicated NN policy whose many weights are initialized randomly does not collect useful experiences to seek its goal, it is not promising to collect good experiences by itself, especially at the beginning of the learning. Therefore, we have to find an appropriate network architecture based on the task's complexity. Although an evolutionary method was applied to the problem of a neural architecture search (Whiteson and Stone, 2006) for tiny problems, experimenters usually manually prepare a learning module with an appropriate network architecture depending on the situation. Furthermore, it is crucial to select an appropriate RL algorithm based on the given task. For instance, two major types of algorithms exist: value-based reinforcement learning and policy search methods, including policy gradient reinforcement learning. Such value-based reinforcement learning as Q-learning (Watkins and Dayan, 1992) and SARSA (Rummery and Niranjan, 1994) learns faster than vanilla policy search methods such as REINFORCE (Williams, 1992) because value-based reinforcement learning exploits the Bellman equation under the Markovian assumption. The policy search methods are robust and find a better stochastic policy even if the state representation is deficient (Kalyanakrishnan and Stone, 2011).

In practice, experimenters test different combinations to select the best one since their appropriate combination is unknown in advance. Moreover, since the sequential testing of these factors is very time-consuming, to eliminate the need for such human hand-tuning, we proposed Cooperative and competitive Learning with Importance Sampling (CLIS) (Uchibe and Doya, 2004, 2005). Here, the agent possesses multiple heterogeneous learning modules and selects an appropriate module based on the task and its experience. We consider a mechanism by which an agent can best utilize its behavioral experiences to train multiple learning modules with different network architecture and learning algorithms. By exploiting task-relevant experiences gathered by suboptimal but fast-learning modules, a complicated module learns faster than when it was trained alone. Unfortunately, CLIS is unstable in learning for several reasons. One is the naive use of importance sampling to compensate for the mismatch in the target and behavior policies. The other is that the original CLIS adopts classical RL algorithms and linear function approximators. In addition, the application of CLIS to robot control is quite limited because it is implicitly assumed that the action is discrete.

To overcome the problems raised by the study of CLIS, this paper proposes Cooperative and competitive Reinforcement And Imitation Learning (CRAIL), which extends CLIS to stabilize learning processes and improve sampling efficiency. Similar to CLIS, CRAIL maintains a set of multiple heterogeneous policies, including hand-coded controllers, and collects samples by a behavior policy constructed by the mixture distribution of the policies. Because the mixing weights are computed by the performance of the module, a better policy is automatically selected based on the learning progress. Then all the modules are trained simultaneously by two objective functions. CRAIL introduces the following two components to CLIS: (1) multiple importance sampling, and (2) policy learning using a combination of temporal difference and behavior cloning loss. Using multiple importance sampling stabilizes the learning process of the policy search methods because the correction factor, which is called the importance-sampling ratio, is upper-bounded. One critical contribution of CRAIL is its introduction of behavior cloning loss as well as temporal difference learning. Based on the learning processes of multiple modules, CRAIL dynamically updates the behavior policy that will be used as the best expert policy. Unlike learning from demonstrations, we can explicitly compute the behavior cloning loss based on a behavior policy, which significantly improves the policy updates. Furthermore, we use modern reinforcement learning algorithms such as entropy-regularized RL because of several advantages described later.

We compare CRAIL with CLIS on four benchmark control tasks supported by the OpenAI gym (Brockman et al., 2016). Experimental results indicate that by exploiting task-relevant episodes generated by suboptimal, but fast-learning modules a complex learning module trained with CRAIL actually learns faster than when it is trained alone. Due to adding the behavior cloning loss, CRAIL learns much faster than CLIS on all the benchmark tasks. In addition, CRAIL effectively transfers samples collected by the fixed hand-coded controller to train policies implemented by neural networks.

2. RELATED WORK

Several reinforcement learning methods with multiple modules have been proposed. Compositional Q-learning (Singh, 1992) selects a learning module with the least TD-error, and Selected Expert Reinforcement Learner (Ring and Schaul, 2011) extends the value function to select a module with better performance. Doya et al. (2002) proposed Multiple Model-based Reinforcement Learning (MMRL), in which each module is comprised of a state prediction model and the module with the least prediction error is selected and trained. These approaches are interpreted as the concept of "Mixture of Experts." In these approaches, the structure of each module is the same and uses the same learning algorithm, while CRAIL enables the use of heterogeneous learning modules that can be trained concurrently. One interpretation is that the modules are spatially distributed in their methods because they change the module based on the current environmental state. On the other hand, CRAIL temporarily distributes the modules because it switches them due to the learning progress.

Some researchers integrated an RL algorithm with hand-coded policies to improve the learning progress in its initial stage. Smart and Kaelbling (2002) proposed an architecture comprised of a supplied control policy and Q-learning. In the first learning phase, a robot was controlled with the supplied control policy developed by a designer. The second learning phase begins to control the robot effectively when the value function is approximated sufficiently. Xie et al. (2018) proposed a similar approach to incorporate *a prior* knowledge, in which Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) and a PID controller are used as off-policy learning and a hand-coded policy, respectively. However, a limitation of their approach is that it uses only one learning module. CRAIL is a more general architecture for incorporating multiple prior knowledge. In addition, it can automatically select an appropriate module depending on the learning progress. Sutton et al. (1999) described the advantages of off-policy learning and proposed a novel framework to accelerate learning by representing policies at multiple levels of temporal abstraction. Although their method assumed a semi-Markov decision problem and AVRL, CLIS can use different learning algorithms.

Our framework can be interpreted as learning from demonstrations. Many previous studies can be found in this field, and some recent studies such as (Gao et al., 2018; Hester et al., 2018; Nair et al., 2018) integrated reinforcement learning with learning from demonstrations by augmenting the objective function. Our framework resembles those methods from the viewpoint of the design of the objective function. The role of the demonstrator is different because our framework's demonstrator is selected from multiple heterogeneous policies based on the learning progress; previous studies assumed that it is stationary and used it to generate a training dataset. Since CRAIL explicitly represents the behavior policy, actions can be easily sampled from it to evaluate the behavior cloning loss.

The most closely related study is Mix & Match (Czarnecki et al., 2018), in which multiple heterogeneous modules are trained in parallel. Mix & Match's basic idea resembles CRAIL, but it does not consider multiple reinforcement learning algorithms; CRAIL adopts three learning algorithms for every module. In addition, Mix & Match uses a mixture of policies and optimizes the mixing weights by a kind of evolutionary computation. Since Mix & Match needs multiple simulators, it is sample-inefficient. The mixing weights are automatically determined in the case of CRAIL.

3. METHODS

3.1. CRAIL's Architecture

We investigate the standard Markov Decision Process (MDP) framework, which is not known by an agent in the model-free RL setting (Sutton and Barto, 1998). An MDP is formulated as follows: (1) $\mathcal{X}$ is the state space and $\boldsymbol{x}_t \in \mathcal{X}$ denotes the state of the environment at time t; (2) $\mathcal{U}$ is the action space and $\boldsymbol{u}_t \in \mathcal{U}$ is the action executed by the agent at time t; (3) $p_e(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u})$ is the state transition probability for $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ and $\boldsymbol{u} \in \mathcal{U}$; (4) $p_0(\boldsymbol{x})$ is the initial state probability; and (5) $r(\boldsymbol{x}, \boldsymbol{u})$ is a reward function. CRAIL has M learning modules as shown in **Figure 1**, and each of which has state value function $V_i(\boldsymbol{x}; \boldsymbol{\psi}_i)$, state-action value function $Q_i(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{\theta}_i)$, and policy $\pi_i(\boldsymbol{u} \mid \boldsymbol{x}; \boldsymbol{\phi}_i)$, where $\boldsymbol{\psi}_i, \boldsymbol{\theta}_i$, and $\boldsymbol{\phi}_i$ are the parameters, respectively. V_i and Q_i are defined as a discounted sum of the rewards given by

$$V_i(\boldsymbol{x}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{x}_t, \boldsymbol{u}_t) \,\middle|\, \boldsymbol{x}_0 = \boldsymbol{x}\right],$$

$$Q_i(\boldsymbol{x}, \boldsymbol{u}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{x}_t, \boldsymbol{u}_t) \,\middle|\, \boldsymbol{x}_0 = \boldsymbol{x}, \boldsymbol{u}_0 = \boldsymbol{u}\right],$$

where $\gamma \in [0, 1)$ is a discount factor that determines the relative weighting of immediate versus later rewards. For simplicity, all the modules share the same sensory-motor system.

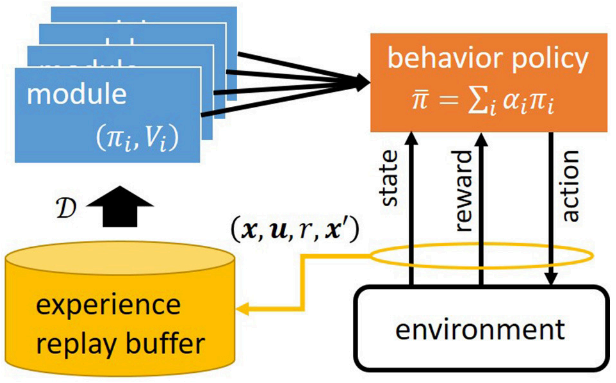


FIGURE 1 | Architecture of Cooperative and Competitive Reinforcement And Imitation Learning (CRAIL).

Algorithm 1 Stepwise CRAIL

```
1:  Initialize all parameters of the learning modules.
2:  Initialize empty replay buffer $\mathcal{D}$.
3:  repeat
4:      $\boldsymbol{x}_0 \sim p_0(\cdot)$                          ▷ Draw an initial state.
5:      for $t = 0, \dots, T-1$ do
6:          $\boldsymbol{u}_t \sim \bar{\pi}(\cdot \mid \boldsymbol{x}_t), \boldsymbol{x}_{t+1}, r_t \sim p_e(\cdot, \cdot \mid \boldsymbol{x}_t, \boldsymbol{u}_t)$.
7:          Add batch data $\{\boldsymbol{x}_t, \boldsymbol{u}_t, r_t, \boldsymbol{x}_{t+1}\}$ to replay buffer $\mathcal{D}$.
8:          for $i = 1, \dots, M$ do              ▷ Update all the modules.
9:              update the parameters by Algorithms 4 or 5.
10:         end for
11:     end for
12: until convergence
```

At each time step t, the agent selects an action based on the following behavior policy:

$$\bar{\pi}(\boldsymbol{u}_t \mid \boldsymbol{x}_t) = \sum_{i=1}^{M} \alpha(i \mid \boldsymbol{x}_t) \pi_i(\boldsymbol{u}_t \mid \boldsymbol{x}_t; \boldsymbol{\phi}_i). \tag{1}$$

Because the state value function evaluates the policy's performance, we use it to determine the mixing weight:

$$\alpha(i \mid \boldsymbol{x}_t) = \frac{\exp(\beta V_i(\boldsymbol{x}_t; \boldsymbol{\psi}_i))}{\sum_{j=1}^{M} \exp(\beta V_i(\boldsymbol{x}_t, \boldsymbol{\psi}_j))}, \tag{2}$$

where β is an inverse temperature. A low β value causes (most of) the equiprobable selection of all the modules, while its high value causes the selection of a module with the highest value when the probability comes closest to one. Inverse temperature β plays an important role at the early stage of learning concerning whether to select optimistic modules that may have large initial values. **Algorithm 1** illustrates an overview of the learning process of stepwise CRAIL. The agent maintains experience replay buffer $\mathcal{D}$ to store state transition $(\boldsymbol{x}, \boldsymbol{u}, r, \boldsymbol{x}')$ by behavior policy $\bar{\pi}$.

As a special case for episodic tasks, we focus on episodic CRAIL, which is basically identical to the original CLIS, as shown

Algorithm 2 Episodic CRAIL/CLIS

```
 1: Initialize all parameters of the learning modules.
 2: Initialize empty replay buffer $\mathcal{D}$.
 3: repeat
 4:     for $k = 1, \dots, K$ do                          ▷ Collect $K$ episodes
 5:         $\boldsymbol{x}_0 \sim p_0(\cdot)$                  ▷ Draw an initial state.
 6:         $i \sim \alpha(\cdot \mid \boldsymbol{x}_0)$                ▷ Select a module.
 7:         for $t = 0, \dots, T-1$ do
 8:             $\boldsymbol{u}_t \sim \pi_i(\cdot \mid \boldsymbol{x}_t), \boldsymbol{x}_{t+1}, r_t \sim p_e(\cdot, \cdot \mid \boldsymbol{x}_t, \boldsymbol{u}_t)$.
 9:         end for
10:         Add batch data $\{i, \boldsymbol{x}_{0:T}, \boldsymbol{u}_{0:T-1}, r_{0:T-1}\}$ to replay buffer $\mathcal{D}$.
11:         for $i = 1, \dots, M$ do              ▷ Update all the modules.
12:             update the parameters by Algorithms 3, 4, or 5.
13:         end for
14:     end for
15: until convergence
```

in **Algorithm 2**. At the beginning of every episode, a module is chosen by Equation (2) to generate a sequence of states, actions, and rewards denoted by

$$h \triangleq [\boldsymbol{x}_1, \boldsymbol{u}_1, r_1, \dots, \boldsymbol{x}_T, \boldsymbol{u}_T, r_T],$$

where T denotes the number of steps called the horizon length. This modification is useful from the viewpoint of numerical stability when a hand-coded deterministic policy is used as domain knowledge. For example, a Central Pattern Generator (CPG) is widely used to generate rhythmic motions like walking without rhythmic sensory inputs (Ijspeert, 2008), but it cannot be represented by policy $\pi_i(\boldsymbol{u} \mid \boldsymbol{x})$ because CPG has internal states that are not observable by other modules. In this case, the module has to cope with partially observable MDP tasks if the experiences generated by the CPG-based controller are used for training.

3.2. Learning Algorithm in Each Module

Similar to CLIS, all the modules learn an optimal policy in parallel on the samples from $\mathcal{D}$ collected by the behavior policy. The learning algorithms used by CRAIL should be able to learn from the experiences gathered by other modules, and therefore, we adopt the following three methods as an off-policy RL algorithm: REINFORCE (Williams, 1992), Soft Actor-Critic (Soft AC) (Haarnoja et al., 2018), and Deterministic Policy Gradient (DPG) (Lillicrap et al., 2016). We modify these algorithms by incorporating behavior loss to update the policy to improve their learning efficiency.

3.2.1. REINFORCE With Importance Sampling

Policy search methods that do not rely on the Bellman optimality equation such as REINFORCE (Williams, 1992) have been reevaluated because of their simplicity and robust performance with non-Markovian tasks (Meuleau et al., 1999). REINFORCE is essentially an on-policy method (Sutton and Barto, 1998) because it estimates the gradient at a particular point in the policy space by acting precisely in the manner of its corresponding policy during learning trials. To use samples collected by the behavior policy, we introduce importance sampling to the REINFORCE algorithm (Meuleau et al., 2001) as an off-policy learning algorithm. Note that REINFORCE is applicable for the episodic CRAIL because it requires a set of sequences as a dataset.

REINFORCE evaluates sequence h by

$$J_i^\pi(\boldsymbol{\phi}_i, h) = R(h) = \sum_{t=1}^{T} \gamma^{t-1} r_t,$$

where $R(h)$ is called the return, which is defined as the discounted sum of rewards along h. To update $\boldsymbol{\phi}_i$, REINFORCE adopts the stochastic gradient ascent method with the gradient given by

$$\frac{\partial J_i^\pi(\boldsymbol{\phi}_i, h)}{\partial \boldsymbol{\phi}_i} = (R(h) - b)\rho_i(h) \sum_{t=1}^{T} \frac{\partial \ln \pi_i(\boldsymbol{u}_t \mid \boldsymbol{x}_t)}{\partial \boldsymbol{\phi}_i}, \tag{3}$$

where b is a baseline parameter for variance reduction and $\rho_i(h)$ is the importance-sampling weight ratio to account for the change in the distribution, defined by

$$\rho_i(h) = \prod_{t=1}^{T} \rho_i(\boldsymbol{x}_t, \boldsymbol{u}_t) = \prod_{t=1}^{T} \frac{\pi_i(\boldsymbol{u}_t \mid \boldsymbol{x}_t)}{\bar{\pi}(\boldsymbol{u}_t \mid \boldsymbol{x}_t)}, \tag{4}$$

under the Markovian assumption. Unlike CLIS, CRAIL uses multiple importance sampling in which the denominator in (4) is the mixture distribution (1) and therefore ρ_i is upper-bounded. Note that Equation (3) is slightly different from the standard expression because the expected value with respect to all possible sequences should be considered to exploit the baseline and importance sampling. We will take expectations later to clarify how the gradient of our method is different from the original one.

Although the gradient estimator (3) is sample-efficient, it is close to zero when π_i is far from $\bar{\pi}$. This situation is often observed at the early stage of learning. To overcome this problem, we introduce the following additional objective function given by the KL divergence between the learning and behavior policies:

$$J_i^{\mathrm{BC}}(\boldsymbol{\phi}_i, \boldsymbol{x}_t) = D_{\mathrm{KL}}(\bar{\pi}(\cdot \mid \boldsymbol{x}_t) \parallel \pi_i(\cdot \mid \boldsymbol{x}_t)). \tag{5}$$

Minimizing (5) is behavior cloning, which is also known as supervised imitation learning. However, our method is more computationally efficient because we can draw samples from $\bar{\pi}$ without interacting through the environment. Consequently, the gradient to train the policy parameter is given by

$$\frac{\partial J_i^\pi(\boldsymbol{\phi}_i)}{\partial \boldsymbol{\phi}_i} = \mathbb{E}_{h\sim\mathcal{D}}\left[\frac{\partial J_i^{\pi,\mathrm{RL}}(\boldsymbol{\phi}_i, \cdot)}{\partial \boldsymbol{\phi}_i}\right] - \eta \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{\partial J_i^{\pi,\mathrm{BC}}(\boldsymbol{\phi}_i, \cdot)}{\partial \boldsymbol{\phi}_i}\right], \tag{6}$$

where η is a positive meta-parameter. When $\eta = 0$, Equation (6) is identical to the original gradient estimator of REINFORCE with importance sampling.

Finally, state value function $V_i(\boldsymbol{x}, \boldsymbol{\psi}_i)$ is also trained with the Monte Carlo method because it is used to construct the behavior

Algorithm 3 REINFORCE with importance sampling and Imitation Learning

```
Require: dataset $\mathcal{D}$
1: Sample a random minibatch of sequences $h$ from $\mathcal{D}$.
2: Evaluate gradient $\partial J_i^{\pi,\mathrm{RL}}/\partial \boldsymbol{\phi}_i$.
3: Sample a random minibatch of states $\boldsymbol{x}$ from $\mathcal{D}$ and $\boldsymbol{u}$ from $\bar{\pi}$, respectively.
4: Evaluate gradient $\partial J_i^{\pi,\mathrm{BC}}/\partial \boldsymbol{\phi}_i$.
5: Update $\boldsymbol{\phi}_i$ by the stochastic gradient ascent method with Equation (6).
6: Update $\boldsymbol{\psi}_i$ by minimizing Equation (7).
```

policy. When the number of sequences in $\mathcal{D}$ is denoted by K, the loss function to optimize the state value function is given by

$$J_i^V(\boldsymbol{\psi}_i) = \frac{1}{2}\sum_{k=1}^{K}\sum_{t=1}^{T}\left(V_i(\boldsymbol{x}_t^k) - Y_t^k\right)^2, \tag{7}$$

where Y_t^k is the target value defined as

$$Y_t^k = \prod_{t'=t}^{T}\rho(\boldsymbol{x}_t^k, \boldsymbol{u}_t^k)\sum_{t'=t}^{T}\gamma^{t'-t}r_t^k.$$

The update rule of the modified REINFORCE with importance sampling is given in **Algorithm** (**3**).

3.2.2. Soft Actor-Critic and Imitation Learning

The original CLIS adopted SARSA (Rummery and Niranjan, 1994) with importance sampling (Precup et al., 2001) as an off-policy value-based reinforcement learning algorithm. An advantage is that the technique called eligibility traces (Sutton and Barto, 1998) can be used to accelerate the speed of learning, and it was experimentally shown that deep SARSA can achieve a comparable performance to DQN even though it does not exploit the method of experience replay and target network (Elfwing et al., 2018). However, SARSA implicitly assumes that action is discrete because the stochastic policy must be derived from the state-action value function. Since we are interested in robot control, action must be continuous. Therefore, we adopt Soft Actor-Critic (Haarnoja et al., 2018) as an off-policy algorithm using the value function. Soft Actor-Critic augments the reward function to replace the max-operator with a differentiable one. The reward function is assumed to be given by the following form:

$$\tilde{r}(\boldsymbol{x}, \boldsymbol{u}) = r(\boldsymbol{x}, \boldsymbol{u}) + \frac{1}{\alpha}\mathcal{H}(\pi_i(\cdot \mid \boldsymbol{x})), \tag{8}$$

where α is a positive meta-parameter and and $\mathcal{H}(\pi(\cdot \mid \boldsymbol{x}))$ is the (differential) entropy of policy π_i. Assuming reward function (8), an optimal state value function satisfies the following Bellman optimality equation:

$$V_i(\boldsymbol{x}) = \max_{\pi_i}\mathbb{E}_{\pi_i}\left[r(\boldsymbol{x}, \boldsymbol{u}) - \frac{1}{\alpha}\ln\pi_i(\boldsymbol{u} \mid \boldsymbol{x}) + \gamma\mathbb{E}_{P_T}\left[V_i(\boldsymbol{x}')\right]\right]. \tag{9}$$

The right hand side of Equation (9) is a constrained optimization problem given by

$$\max_{\pi_i}\int \mathrm{d}\boldsymbol{u}\pi_i(\boldsymbol{u} \mid \boldsymbol{x})\left[r(\boldsymbol{x}, \boldsymbol{u}) - \frac{1}{\alpha}\ln\pi_i(\boldsymbol{u} \mid \boldsymbol{x}) + \gamma\mathbb{E}_{P_T}\left[V_i(\boldsymbol{x}')\right]\right],$$

subject to $\int \mathrm{d}\boldsymbol{u}\pi_i(\boldsymbol{u} \mid \boldsymbol{x}) = 1$. In this case, we can analytically maximize the right hand side of Equation (9) by a method with Lagrange multipliers. Consequently, the optimal state value function can be represented by

$$V_i(\boldsymbol{x}) = \frac{1}{\alpha}\ln\int \mathrm{d}\boldsymbol{u}\left[\exp(\alpha Q_i(\boldsymbol{x}, \boldsymbol{u}))\right], \tag{10}$$

and the corresponding optimal policy can be derived:

$$\pi_i(\boldsymbol{u} \mid \boldsymbol{x}) = \frac{\exp\left(\alpha Q_i(\boldsymbol{x}, \boldsymbol{u})\right)}{\exp(\alpha V_i(\boldsymbol{x}))}, \tag{11}$$

where state-action value function $Q(\boldsymbol{x}, \boldsymbol{u})$ is defined by

$$Q_i(\boldsymbol{x}, \boldsymbol{u}) = r(\boldsymbol{x}, \boldsymbol{u}) + \gamma\mathbb{E}_{P_T}\left[V_i(\boldsymbol{x}')\right]. \tag{12}$$

Note that the right hand side of Equation (10) uses the log-sum-exp operator if the action is discrete, and it is characterized as the "soft" max operator.

The learning algorithm of the Soft Actor-Critic is derived from Equations (10)–(12). Since Equation (12) corresponds to the Bellman optimality equation regarding the state-action value function, it can be used to train parameter $\boldsymbol{\theta}_i$ by minimizing the soft Bellman residual for all possible $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ in buffer $\mathcal{D}$:

$$J_i^Q(\boldsymbol{\theta}_i, \boldsymbol{x}, \boldsymbol{u}, r, \boldsymbol{x}') = \frac{1}{2}\left\{Q_i(\boldsymbol{x}, \boldsymbol{u}) - \left(r + \gamma V_i(\boldsymbol{x}'; \bar{\boldsymbol{\psi}}_i)\right)\right\}^2,$$

where $V_i(\boldsymbol{x}, \bar{\boldsymbol{\psi}}_i)$ and $\bar{\boldsymbol{\psi}}_i$ respectively denote the target state value network and an exponentially moving average of the parameter vector, which stabilizes the learning used in DQN (Mnih et al., 2015). Consequently, the loss function for training $\boldsymbol{\theta}_i$ is given by

$$J_i^Q(\boldsymbol{\theta}_i) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{u},r,\boldsymbol{x}')\sim\mathcal{D}}\left[J_i^Q(\boldsymbol{\theta}_i, \cdot, \cdot, \cdot, \cdot)\right], \tag{13}$$

where $(\boldsymbol{x}, \boldsymbol{u}, r, \boldsymbol{x}') \sim \mathcal{D}$ means that the transition data are drawn from $\mathcal{D}$.

When the action is discrete, the optimal policy and the state value function can be easily computed from the state-action value function. However, it is intractable in the case of continuous action because Equation (10) needs to evaluate the integral in action space. Therefore, Haarnoja et al. (2018) recommended that the state value function and policy also be separately approximated. Based on the relation (11), the approximation error of the state value function at state $\boldsymbol{x}$ is given by

$$J_i^V(\boldsymbol{\psi}_i, \boldsymbol{x}) = \frac{1}{2}\left\{V_i(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{u}\sim\pi_i}\left[Q_i(\boldsymbol{x}, \cdot) - \frac{1}{\alpha}\ln\pi(\cdot \mid \boldsymbol{x})\right]\right\}^2,$$

where the expectation is numerically computed through a Monte Carlo simulation. The loss function for training $\boldsymbol{\psi}_i$ is given by

$$J_i^V(\boldsymbol{\psi}_i) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[J_i^V(\boldsymbol{\psi}_i, \cdot)\right]. \tag{14}$$

Algorithm 4 Soft Actor-Critic and Imitation Learning

```
Require: dataset $\mathcal{D}$, inverse temperature $\eta$, decay rate $\tau$.
1: Sample a random minibatch of transitions $(\boldsymbol{x}, \boldsymbol{u}, r, \boldsymbol{x}')$ from $\mathcal{D}$.
2: Evaluate gradient $\partial J_i^Q / \partial \boldsymbol{\theta}_i$ and update $\boldsymbol{\theta}_i$ by stochastic gradient descent.
3: Sample a random minibatch of states $\boldsymbol{x}$ from $\mathcal{D}$ and $\boldsymbol{u}$ from $\pi_i$, respectively.
4: Evaluate gradient $\partial J_i^V / \partial \boldsymbol{\psi}_i$ and update $\boldsymbol{\psi}_i$ by the stochastic gradient descent.
5: Sample a random minibatch of states $\boldsymbol{x}$ from $\mathcal{D}$ and $\boldsymbol{u}$ from $\bar{\pi}$, respectively.
6: Evaluate gradient $\partial J_i^\pi / \partial \boldsymbol{\phi}_i$ and update $\boldsymbol{\phi}_i$ by the stochastic gradient descent.
7: Update the parameter of the target network by $\bar{\boldsymbol{\psi}}_i \leftarrow \tau \bar{\boldsymbol{\psi}}_i + (1-\tau)\boldsymbol{\psi}_i$.
```

In the same way, policy parameter $\boldsymbol{\theta}_i$ is trained by minimizing the Kullback-Leibler (KL) divergence between the left and right hand sides of Equation (11):

$$J_i^{\pi,\mathrm{RL}}(\boldsymbol{\phi}_i, \boldsymbol{x}) = D_{\mathrm{KL}}\left(\pi_i(\cdot \mid \boldsymbol{x}) \parallel \frac{\exp(\alpha Q_i(\boldsymbol{x}, \cdot))}{\exp(\alpha V_i(\boldsymbol{x}))}\right), \tag{15}$$

where we need samples drawn from π_i to evaluate the KL divergence. In addition to the KL divergence, we introduce the behavior cloning loss defined as the KL divergence between the learning and behavior policies:

$$J_i^{\pi,\mathrm{BC}}(\boldsymbol{\phi}_i, \boldsymbol{x}) = D_{\mathrm{KL}}(\bar{\pi}(\cdot \mid \boldsymbol{x}) \parallel \pi_i(\cdot \mid \boldsymbol{x})). \tag{16}$$

Consequently, the loss function for training $\boldsymbol{\phi}_i$ is given by

$$J_i^{\pi}(\boldsymbol{\phi}_i) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[J_i^{\pi,\mathrm{RL}}(\boldsymbol{\phi}_i, \cdot) + \eta J_i^{\pi,\mathrm{BC}}(\boldsymbol{\phi}_i, \cdot)\right], \tag{17}$$

where η is a positive meta-parameter. When $\eta = 0$, Equation (17) is identical to the original update rule of Soft Actor-Critic. Note that Information projection (I-projection) is used in Equation (15), and Moment projection (M-projection) is used in Equation (16) (Kober et al., 2013). Although in principle we can select any projection, we believe that Equation (16) is appropriate for the behavior cloning loss because it is averaged over several modes of the policy. In addition, Equation (15) is appropriate because it concentrates on a single mode. π_i is usually implemented by a Gaussian policy with a single mode, but $\exp(\alpha Q_i(\boldsymbol{x}, \cdot))/\exp(\alpha V_i(\boldsymbol{x}))$ may have multiple modes. The update rule of the modified Soft Actor-Critic is given by **Algorithm 4**.

3.2.3. Deterministic Policy Gradient and Imitation Learning

Deterministic Policy Gradient (DPG) (Silver et al., 2014) and its deep version (Lillicrap et al., 2016) are a well-known off-policy reinforcement learning algorithm that can handle continuous actions. Unlike Soft Actor-Critic, DPG does not approximate the state value function. The policy network can also be simplified significantly because it does not need to approximate a continuous probability density function.

Algorithm 5 Deterministic Policy Gradient and Imitation Learning

```
Require: dataset $\mathcal{D}$, inverse temperature $\eta$, decay rate $\tau$.
1: Sample a random minibatch of transitions $(\boldsymbol{x}, \boldsymbol{u}, r, \boldsymbol{x}')$ from $\mathcal{D}$.
2: Evaluate gradient $\partial J_i^Q / \partial \boldsymbol{\theta}_i$ and update $\boldsymbol{\theta}_i$ by the stochastic gradient descent.
3: Sample a random minibatch of states $\boldsymbol{x}$ from $\mathcal{D}$.
4: Evaluate gradient $\partial J_i^\pi / \partial \boldsymbol{\phi}_i$ and update $\boldsymbol{\phi}_i$ by the stochastic gradient descent.
5: Update the parameter of the target network by $\bar{\boldsymbol{\theta}}_i \leftarrow \tau \bar{\boldsymbol{\theta}}_i + (1-\tau)\boldsymbol{\theta}_i$.
```

The loss function to train Q_i in DPG resembles that in Soft Actor-Critic and is given by Equation (13) whose $J_i^Q(\boldsymbol{\theta}_i, \boldsymbol{x}, \boldsymbol{u}, r, \boldsymbol{x}')$ is replaced with the following equation:

$$J_i^Q(\boldsymbol{\theta}_i, \boldsymbol{x}, \boldsymbol{u}, r, \boldsymbol{x}') = \frac{1}{2}\left\{Q_i(\boldsymbol{x}, \boldsymbol{u}) - (r + \gamma Q_i(\boldsymbol{x}', \pi_i(\boldsymbol{x}'); \bar{\boldsymbol{\theta}}_i))\right\}^2,$$

where $\bar{\boldsymbol{\theta}}_i$ denotes an exponentially moving average of the parameter vector of the target state-action value network and π_i is a deterministic policy that maps $\boldsymbol{x}$ to $\boldsymbol{u}$. DPG evaluates the policy gradient at state $\boldsymbol{x}$ by

$$\frac{\partial J_i^{\pi,\mathrm{RL}}(\boldsymbol{\phi}_i, \boldsymbol{x})}{\partial \boldsymbol{\phi}_i} = \left.\frac{\partial Q_i(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}=\pi_i(\boldsymbol{x})} \frac{\partial \pi_i(\boldsymbol{x})}{\partial \boldsymbol{\phi}_i}.$$

As a result, the policy gradient with behavior cloning loss is computed by

$$\frac{\partial J_i^{\pi}}{\partial \boldsymbol{\phi}_i} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{\partial J_i^{\pi,\mathrm{RL}}(\boldsymbol{\phi}_i, \cdot)}{\partial \boldsymbol{\phi}_i} - \eta \frac{\partial J_i^{\pi,\mathrm{BC}}(\boldsymbol{\phi}_i, \cdot)}{\partial \boldsymbol{\phi}_i}\right],$$

where $J_i^{\pi,\mathrm{BC}}$ is the same function used by the modified Soft Actor-Critic explained in section 3.2.3. The state value function is simply computed by

$$V_i(\boldsymbol{x}) = Q_i(\boldsymbol{x}, \pi_i(\boldsymbol{x})).$$

The update rule of the modified DPG is given by **Algorithm 5**. One limitation of DPG is that it does not have an explicit exploration mechanism because policy π_i represents a deterministic function. Therefore, DPG usually introduces a behavior policy that is implemented by an Ornstein-Uhlenbech process (Lillicrap et al., 2016). On the other hand, CRAIL's behavior policy is dynamically constructed by mixing all of the component policies. When DPG is selected as a learning algorithm of CRAIL, at least one learning module with a stochastic policy should be added to promote exploration and discourage premature convergence.

4. EXPERIMENTS

4.1. Comparison of CRAIL and CLIS

To investigate how CRAIL improves the learning speed, we conducted several computer simulations with four MuJoCo-simulated (Todorov et al., 2012) benchmark tasks: Hopper-v2, Half-Cheetah-v2, Walker2d-v2, and Ant-v2, all of which were provided by the OpenAI gym (Brockman et al., 2016) (**Figure 2**). Hopper-v2 is a planar monopod, and Walker2d-v2 and HalfCheetah-v2 are planar biped robots. Ant-v2 is a quadruped robot that can move around a three-dimensional environment. The observation and action spaces are shown in **Table 1**, where the observation vector is used as a state vector. The goal is to move forward as quickly as possible, and the reward function is given by $r(\boldsymbol{x}, \boldsymbol{u}) = v_x - c\|\boldsymbol{u}\|_2^2$, where v_x is the forward velocity and c is a robot-dependent constant. See the supplementary materials of Duan et al. (2016) for the task specifications.

We prepared two function approximators, Neural Network (NN) and normalized Radial Basis Function (RBF), and **Table 2** shows their network architectures. For example, the module using the RBF networks represents V_i by 64 normalized radial basis functions by

$$V_i(\boldsymbol{x}; \boldsymbol{\psi}_i) = \sum_{j=1}^{N_i} \psi_{i,j} b_{i,j}(\boldsymbol{x}),$$

where N_i and $\psi_{i,j}$ respectively denote the number of basis functions and the j-th element of $\boldsymbol{\psi}_i$ and $b_{i,j}(\boldsymbol{x})$ is the basis function defined by

$$b_{i,j}(\boldsymbol{x}) = \frac{a_{i,j}(\boldsymbol{x})}{\sum_{j'=1}^{N_i} a_{i,j'}(\boldsymbol{x})}, \quad a_{i,j} = \exp\left(-\|\boldsymbol{s}_{i,j}^\top(\boldsymbol{x} - \boldsymbol{c}_{i,j})\|_2^2\right),$$

where $a_{i,j}$ is a Gaussian activation function with parameters $\boldsymbol{s}_{i,j}$ and $\boldsymbol{c}_{i,j}$. Since $\boldsymbol{s}_{i,j}$ and $\boldsymbol{c}_{i,j}$ were determined by a heuristic

TABLE 2 | Network architectures of approximator in the first and the second experiments: For example, RBF module approximates Q_i by 64 basis functions, and NN module approximates two-layer feed-forward neural network consisting of (400, 300) hidden units.

Approximator	V	Q	π
RBF	(64)	(64)	(64)
NN	(64, 64)	(400, 300)	(400, 300)

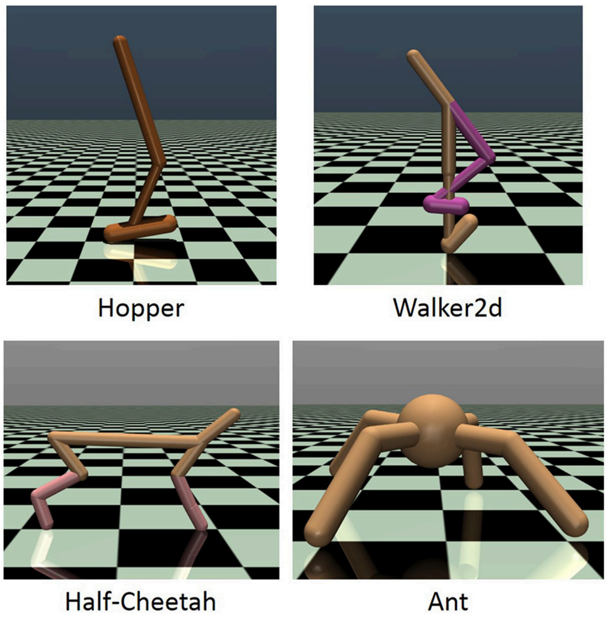


FIGURE 2 | MuJoCo-simulated environments: Hopper-v2, Walker2D-v2, Half-Cheetah-v2, and Ant-v2.

TABLE 1 | Environments used in experiments and their state and action spaces.

Environment	Observation space	Action space
Ant-v2	$\mathbb{R}^{111}$	$[-1.0, 1.0]^8$
HalfCheetah-v2	$\mathbb{R}^{17}$	$[-1.0, 1.0]^6$
Hopper-v2	$\mathbb{R}^{11}$	$[-1.0, 1.0]^3$
Walker2d-v2	$\mathbb{R}^{17}$	$[-1.0, 1.0]^6$

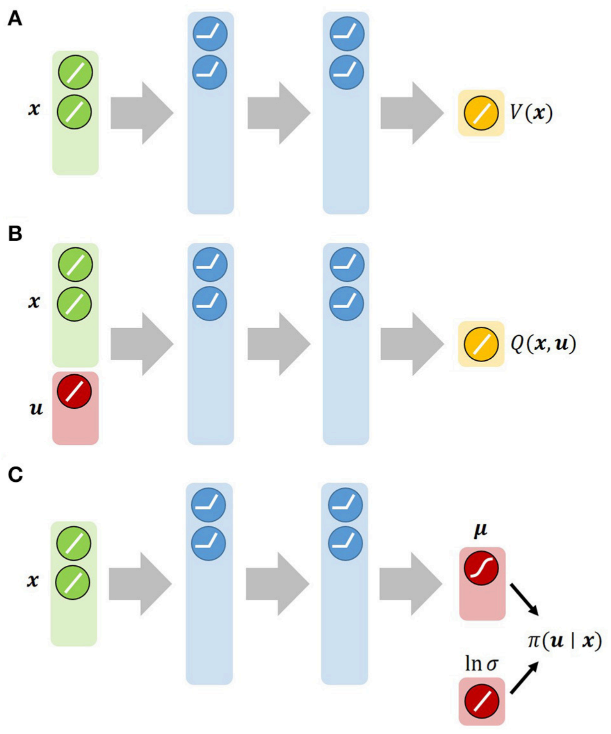


FIGURE 3 | Architectures of neural networks used by Soft Actor-Critic: **(A)** State value function network. **(B)** State-action value function network. **(C)** Gaussian-policy network. We approximate both V and Q with feed-forward neural networks. π is approximated by a Gaussian policy: $\pi(\boldsymbol{u} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{u} \mid \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$, where the mean $\boldsymbol{\mu}$ is given by a neural network and the log-standard deviation $\ln\sigma$ is parameterized by a global vector independent of the state.

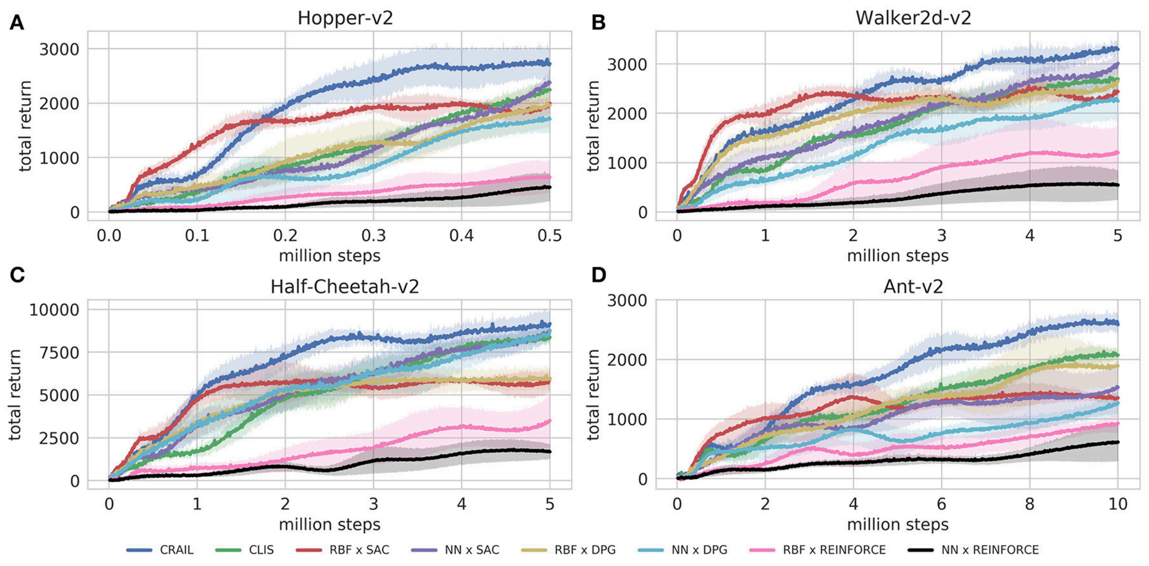


FIGURE 4 | Training curves on continuous control benchmarks: Performance was evaluated by cumulative rewards in 10 episodes for each learning module.

rule (Morimoto and Doya, 2001), V_i is interpreted as a linear neural network. Therefore, the module with the RBF networks is expected to learn faster than that with the nonlinear neural networks. **Figure 3** represents the architectures that approximate π_i, V_i and Q_i needed by the Soft Actor-Critic. Each was implemented by a feed-forward neural network with a Rectified Linear Unit (ReLU) as a nonlinear activation function of the hidden layers. In the first experiment, we chose three learning algorithms, Soft Actor-Critic, Deterministic Policy Gradient, and REINFORCE with importance sampling. We prepared $2 \times 3 = 6$ modules as a result. To apply **Algorithm 3** to this non-episodic task, the horizon length T is set to 300.

CRAIL was given the above six modules for parallel training. We also tested the six modules separately in addition to CLIS as baseline performances, where CLIS also used multiple importance sampling instead of an independent type because the original CLIS worked very poorly due to the unboundedness of the importance-sampling weight ratio. Note that the original CLIS selects one learning module at the beginning of each episode, and utilizes a truncated importance sampling ratio given by

$$\hat{\rho}_i(h) = \min\left(\prod_{t=1}^{T} \frac{\pi_i(\boldsymbol{u}_t \mid \boldsymbol{x}_t)}{\pi_{\text{selected}}(\boldsymbol{u}_t \mid \boldsymbol{x}_t)}, C\right),$$

where π_{selected} is the policy of the selected module and C is a positive constant determined by the experimenters. Although $\hat{\rho}_i(h)$ is upper-bounded, it is not trivial to tune C in practice. In addition, CLIS does not consider behavior cloning loss. Therefore, CLIS evaluated in the experiments uses Equation (4) as the importance weight. In this case, CLIS is identical to CRAIL with $\eta = 0$. Each method was evaluated in ten simulation runs, each of which was comprised of 2,000 episodes.

Figure 4 shows the learning performance of CRAIL, CLIS, and the six component modules, and we found that CRAIL learned faster than CLIS and the six modules trained separately on all the benchmark tasks. On the other hand, the learning performance of CLIS resembled that of the NN × SAC module. The RBF × SAC module showed the best learning curves on all the tasks at the early stage of learning, but its performance saturated before reaching a sufficient level because the normalized RBF networks could not precisely approximate the value functions and the policy as well as the neural networks. On the contrary, the NN policies trained by SAC or DPG learned very slowly, and their performance was much worse than RBF × SAC at the early stage of learning. The modules trained by REINFORCE needs a set of sequences, and therefore, they learned slower than the actor-critic methods such as DPG and Soft AC. As a result, the REINFORCE modules achieved worse performance, and the probabilities remained low during learning. **Figures 5A,B** respectively show the mixing weights $\{\alpha_i\}_{i=1}^{6}$ during the learning of Ant-v2 computed by CRAIL and CLIS. The probability of selecting the RBF × SAC module increased rapidly at the early stage of learning in both cases. However, CRAIL tended to gradually select the NN × SAC module after about four million steps, and CLIS continued to choose the RBF × SAC module's policy most frequently until about six million steps.

4.2. Adaptation to Changes in the Environment

Next, we experimentally tested the capability of adaptation to changes in the environment by changing the mass of the body of HalfCheetah-v2 from 6.36 (original) to 6.36 × 3 [kg] at the 5 millionth step. In this experiment, both CRAIL and CLIS possessed the same six learning modules used in the previous experiment. Each method was evaluated in ten simulation runs, each of which was comprised of 2,000 episodes.

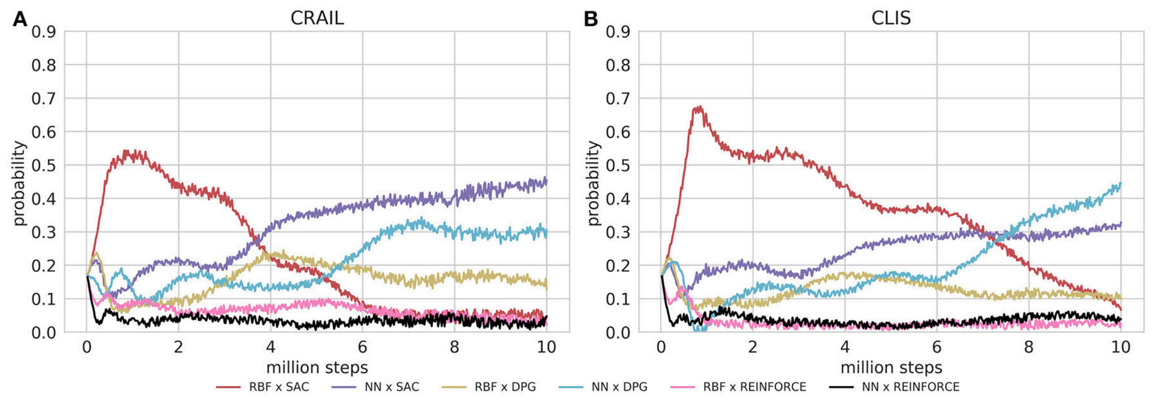


FIGURE 5 | Probabilities for changing learning modules during learning process: **(A)** Results obtained by CLIS architecture. **(B)** Results obtained by CL without importance sampling.

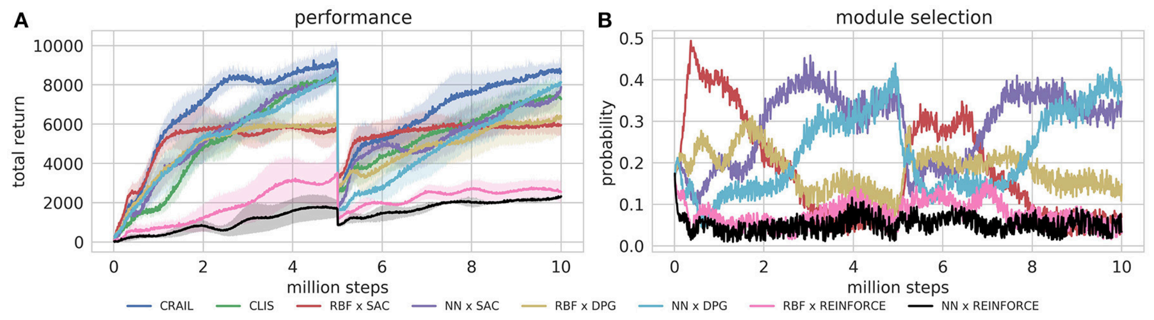


FIGURE 6 | Training curves on episodic Half-Cheetah-v2 task, in which body's mass was changed at 5 millionth step.

Figures 6A,B respectively show the cumulative rewards and module selection probability in each step. Note that the first half of **Figure 6A** is identical to **Figure 4C**. When the mass was changed at 5 millionth steps, the performance of the CRAIL, CLIS, and NN policies decreased significantly. However, the RBF policies maintained the pole without considerable deterioration in performance compared with the NN policies because the number of weights was smaller. In other words, the performances of the NN policies deteriorated drastically because their policies were fine-tuned for a particular weight. Therefore, the probability of selecting RBF × SAC increased temporarily from about 5 to 6.5 million steps. CRAIL prevented the body from falling and trained NN × SAC and NN × DPG by appropriately selecting RBF × SAC, as shown in **Figure 6B**.

4.3. Introducing a Fixed Policy

To investigate how CRAIL exploits a deterministic stationary policy, we added a CPG-based policy as prior knowledge to control HalfCheetah-v2 because periodic motion is quite useful to generate walking behaviors and many previous studies exist (Ijspeert, 2008) in this field. Since CRAIL uses multiple importance sampling, it is straightforward to use the deterministic policy as one of the sampling policies. Note that the CPG-based policy has internal states because the oscillator is implemented by a differential equation. Therefore, we selected the REINFORCE algorithm with importance sampling described in section 3.2.1 and **Algorithm 2** in this experiment because the evaluation of deterministic policies with internal states is difficult in stepwise update rules.

TABLE 3 | Network architectures of approximator in the third experiments: We denote the hidden layer sizes of a two-layer feedforward neural network as *(N, M)*.

Approximator	**V**	π
BASE	(64, 64)	(64, 64)
WIDE	(64, 64)	(400, 300)
DEEP	(64, 64)	(100, 50, 25)

For example, the WIDE module approximates V_i and π_i by (64, 64) and (400, 300), respectively.

As learning modules, we prepared three network architectures that are commonly seen in the literature (Henderson et al., 2018) as shown in **Table 3** to implement a stochastic policy. We used a ReLU nonlinear activation function. Note that the REINFORCE algorithm does not need Q_i. In addition, a deterministic stationary policy based on central pattern generators was prepared as prior knowledge, which was implemented by the modified Hopf oscillator (Uchibe and Doya, 2014). Since CRAIL uses multiple importance sampling, it is straightforward to use the deterministic policy as one of the sampling policies.

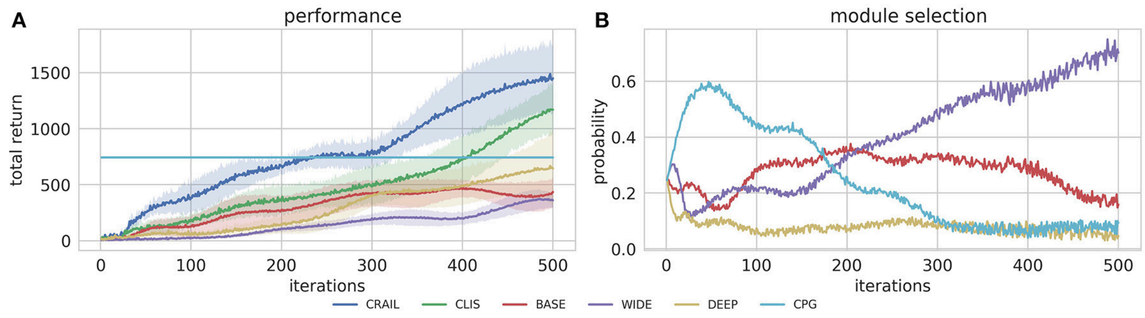


FIGURE 7 | Training curves on Half-Cheetah-v2 task, in which fixed stationary policy was used as prior knowledge.

In addition to evaluate the CRAIL's performance, we tested the four modules separately. **Figure 7A** shows that CRAIL learned much faster than the component modules trained alone. Since REINFORCE learns very slowly due to its simplicity (Duan et al., 2016), 500 iterations were insufficient to overcome the CPG-based controller. **Figure 7B** shows the mixing weights during the learning computed by CRAIL. The probability of selecting the CPG-controller module increased rapidly at the early stage of learning. Then, CRAIL tended to select the BASE module and the probability of selecting it was the highest among the NN modules from about 90 to 170 iterations. Finally, the WIDE module was frequently selected at the later stage of learning. The DEEP module trained alone achieved the highest performance among the three neural network policies. However, the probability of selecting it remained low during learning. Note that the original CLIS cannot utilize the deterministic policy because the importance weight ratio becomes infinity.

5. DISCUSSION

This paper proposed modular reinforcement learning (CRAIL), which collects task-relevant samples using multiple heterogeneous policies. One interesting feature of CRAIL is that a complex RL system can learn faster with the help of a simple RL system that cannot achieve the best performance. Experimental results also suggested that CRAIL efficiently adapted to changes in the learning conditions because it automatically selected simple modules with fewer parameters.

CRAIL implicitly assumes that state value functions are not initialized optimistically. Suppose that the reward function is always non-positive, and the state value functions are initialized to zero. In this case, some modules that are not selected by Equation (1) may have V values that are consistently higher than the selected modules. In this case, CRAIL selects the worst module if the inverse temperature is not tuned appropriately. As one possible extension to overcome this difficulty, the mixing weights are also trained by reinforcement learning in which the value functions are used as priors.

In the current implementation, since all the learning modules are prepared in advance CRAIL cannot obtain good performance if all of them are inappropriate for the given task. To design appropriate learning modules, we need to develop a mechanism to add or delete learning modules based on the selection probabilities calculated by Equation (1). If a simple learning module has a low probability for a long time, it can be replaced by a complicated module. This allows CRAIL to flexibly test heterogeneous modules without increasing computational costs. To overcome this problem, we consider an asynchronous version of the algorithms.

We did not address the effects of computational costs on the learning modules. Updating the parameters of the RBF networks was accomplished considerably faster than for the deep neural networks, but the modules with the RBF networks had to wait until the modules with deep neural networks completed their computations. In general, the sampling rate significantly affects the original performance of a robot. For example, the robot should reduce its moving speed when it uses a complex module. However, the effects of the differences in sampling rates have not been scrutinized.

One interesting future topic is the use of multiple meta-parameters. CRAIL has some meta-parameters used in an RL system, and their settings, such as the learning rate, the inverse temperature that controls the randomness in action selection, and the discount factor for future reward prediction, are crucial to perform a task successfully. A possible scenario is that when a small discount factor can be used in the initial learning process, a module with a larger discount factor can be selected as the learning progresses. We have not yet identified the tasks and situations in which different discount factors play an important role for accelerating the learning speed, but in the future we will seek good examples for this topic.

AUTHOR CONTRIBUTIONS

EU conceived, designed the research, performed the experiment, analyzed its results, and wrote the paper.

FUNDING

This work is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO) and JSPS KAKENHI Grant Numbers JP16K12504 and JP17H06042.

REFERENCES

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). OpenAI Gym [preprint]. *arXiv:1606.01540*.

Czarnecki, M. W., Jayakumar, S. M., Jaderberg, M., Hasenclever, L., Teh, Y. W., Osindero, S., et al. (2018). "Mix & match - Agent curricula for reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning* (Stockholm), 1087–1095.

Doya, K., Samejima, K., Katagiri, K., and Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Comput.* 14, 1347–1369. doi: 10.1162/089976602753712972

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the 33rd International Conference on Machine Learning* (New York, NY), 1329–1338.

Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Netw*. doi: 10.1016/j.neunet.2017.12.012. [Epub ahead of print].

Gao, Y., Xu, H., Lin, J., Yu, F., Levine, S., and Darrell, T. (2018). "Reinforcement learning from imperfect demonstrations," in *ICLR 2018 Workshop* (Vancouver, BC).

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). "Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning* (Stockholm), 1861–1870.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). "Deep reinforcement learning that matters," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (New Orleans, LA).

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., et al. (2018). "Deep Q-learning from demonstrations," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (New Orleans, LA).

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Netw.* 21, 642–653. doi: 10.1016/j.neunet.2008.03.014.

Kalyanakrishnan, S., and Stone, P. (2011). Characterizing reinforcement learning methods through parameterized learning problems. *Mach. Learn.* 84, 205–247. doi: 10.1007/s10994-011-5251-x

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: a survey. *Int. J. Robot. Res.* 32, 1238–1274. doi: 10.1177/0278364913495721

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2016). "Continuous control with deep reinforcement learning," in *Proceedings of International Conference on Learning Representations* (San Juan).

Meuleau, N., Kim, K.-E., Kaelbling, L. P., and Cassandra, A. R. (1999). "Solving POMDPs by searching the space of finite policies," in *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence* (Stockholm), 417–426.

Meuleau, N., Peshkin, L., and Kim, K.-E. (2001). *Exploration in Gradient-Based Reinforcement Learning*. Technical report, Technical Report 2001-003, Cambridge, MA: MIT.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi: 10.1038/nature14236

Morimoto, J., and Doya, K. (2001). Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robot. Auton. Syst.* 36, 37–51. doi: 10.1016/S0921-8890(01)00113-0

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). "Overcoming exploration in reinforcement learning with demonstrations," in *Proceedings of IEEE International Conference on Robotics and Automation* (Brisbane, QLD).

Precup, D., Sutton, R. S., and Dasgupta, S. (2001). "Off-policy temporal-difference learning with function approximation," in *Proceedings of the 18th International Conference on Machine Learning* (Williamstown, MA).

Ring, M., and Schaul, T. (2011). "Q-error as a selection mechanism in modular reinforcement-learning systems," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence* (Barcelona), 1452–1457.

Rummery, G., and Niranjan, M. (1994). *On-Line Q-Learning Using Connectionist Systems*. Technical report, Technical Report CUED/F-INFENG/TR 166, Engineering Department Cambridge University.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature* 529, 484–489. doi: 10.1038/nature1696

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning* (Bejing), 387–395.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of Go without human knowledge. *Nature* 550, 354–359. doi: 10.1038/nature24270

Singh, S. P. (1992). Transfer of learning by composing solution of elemental sequential tasks. *Mach. Learn.* 8, 323–340.

Smart, W. D., and Kaelbling, L. P. (2002). "Effective reinforcement learning for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation* (Washington, DC), 3404–3410.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning*. Cambridge, MA: MIT Press.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.* 112, 181–211.

Todorov, E., Erez, T., and Tassa, Y. (2012). "MuJoCo: a physics engine for model-based control," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* (Vilamoura), 5026–5033.

Uchibe, E., and Doya, K. (2004). "Competitive-cooperative-concurrent reinforcement learning with importance sampling," in *Proceedings of the Eighth International Conference on Simulation of Adaptive Behavior: From Animals to Animats 8* (Los Angeles, CA), 287–296.

Uchibe, E., and Doya, K. (2005). "Reinforcement learning with multiple heterogeneous modules: a framework for developmental robot learning," in *Proceedings of the 4th IEEE International Conference on Development and Learning* (Osaka), 87–92.

Uchibe, E., and Doya, K. (2014). "Combining learned controllers to achieve new goals based on linearly solvable MDPs," in *Proceedings of the IEEE International Conference on Robotics and Automation* (Hong Kong), 5252–5259.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learn.* 8, 279–292.

Whiteson, S., and Stone, P. (2006). Evolutionary function approximation for reinforcement learning. *J. Mach. Learn. Res.* 7, 877–917. Available online at: http://www.jmlr.org/papers/v7/whiteson06a.html

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* 8, 229–256.

Xie, L., Wang, S., Rosa, S., Markham, A., and Trigoni, N. (2018). "Learning with training wheels : speeding up training with a simple controller for deep reinforcement learning," in *Proceedings of IEEE International Conference on Robotics and Automation* (Brisbane, QLD).

Conflict of Interest Statement: The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 01 October 2018
doi: 10.3389/frobt.2018.00110

Stage-Wise Learning of Reaching Using Little Prior Knowledge

François de La Bourdonnaye[1]*, *Céline Teulière*[1], *Jochen Triesch*[2] *and Thierry Chateau*[1]

[1] CNRS, SIGMA Clermont, Institut Pascal, Université Clermont Auvergne, Clermont-Ferrand, France, [2] Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany

In some manipulation robotics environments, because of the difficulty of precisely modeling dynamics and computing features which describe well the variety of scene appearances, hand-programming a robot behavior is often intractable. Deep reinforcement learning methods partially alleviate this problem in that they can dispense with hand-crafted features for the state representation and do not need pre-computed dynamics. However, they often use prior information in the task definition in the form of shaping rewards which guide the robot toward goal state areas but require engineering or human supervision and can lead to sub-optimal behavior. In this work we consider a complex robot reaching task with a large range of initial object positions and initial arm positions and propose a new learning approach with minimal supervision. Inspired by developmental robotics, our method consists of a weakly-supervised stage-wise procedure of three tasks. First, the robot learns to fixate the object with a 2-camera system. Second, it learns hand-eye coordination by learning to fixate its end-effector. Third, using the knowledge acquired in the previous steps, it learns to reach the object at different positions and from a large set of initial robot joint angles. Experiments in a simulated environment show that our stage-wise framework yields similar reaching performances, compared with a supervised setting without using kinematic models, hand-crafted features, calibration parameters or supervised visual modules.

Keywords: deep reinforcement learning, weakly-supervised, stage-wise learning, manipulation robotics, hierarchical learning

OPEN ACCESS

Edited by:
Vieri Giuliano Santucci, Istituto di Scienze e Tecnologie Della Cognizione (ISTC), Italy

Reviewed by:
Kathryn Elizabeth Kasmarik, University of New South Wales Canberra, Australia
Carlos Maestre, Université Pierre et Marie Curie, France

****Correspondence:***
François de La Bourdonnaye
francois.de_la_bourdonnaye@uca.fr

Specialty section:
This article was submitted to Computational Intelligence, a section of the journal Frontiers in Robotics and AI

Received: *25 June 2018*
Accepted: *03 September 2018*
Published: *01 October 2018*

Citation:
de La Bourdonnaye F, Teulière C, Triesch J and Chateau T (2018) Stage-Wise Learning of Reaching Using Little Prior Knowledge. Front. Robot. AI 5:110. doi: 10.3389/frobt.2018.00110

1. INTRODUCTION

In manipulation robotics, various tasks cannot be programmed by hand because dynamics is hard to compute or/and hand-crafted features do not describe well enough the variety of scene appearances. Deep reinforcement learning tackles both of these issues in that features are automatically computed by optimization and dynamics is not required (Levine et al., 2016; Gu et al., 2017; Riedmiller et al., 2018). In manipulation robotics, the success of a task is often defined by a sparse reward (i.e., a positive signal is given to the robot only if the full task is successfully completed) in a high-dimensional state space, which makes learning slow since in some high-dimensional robotics tasks, it is very unlikely to get a first success when the initial states are far from the targeted ones. Although the use of several agents in parallel has shown good performances with a sparse only reward (Levine et al., 2017), it requires expensive resources and materials as well as a simplified action space which are not always possible to get. Provided an expert knowledge is available, learning by demonstration (Kober and Peters, 2009; Nair et al., 2017; Sermanet et al., 2017) can also be used to guide the robot to sparse-reward areas. But it requires prior knowledge on the optimal/sub-optimal behavior for a specific task.

An alternative solution consists of using shaping rewards. They allow to guide the exploration of the agent toward goal state areas, i.e., the probability of receiving sparse rewards is increased. Using shaping rewards leads to two main issues. First, they can lead to sub-optimal policies (Popov et al., 2017) by biasing the exploration process, e.g., the solution specified by the reward function may not be optimal. Second, they generally require tedious engineering work or other forms of supervision. For instance, for manipulation tasks such as block stacking, reaching, door pushing or pulling (Deisenroth et al., 2011; Chebotar et al., 2017; Ghadirzadeh et al., 2017; Gu et al., 2017; Tsurumine et al., 2017), an informative reward is computed based on a distance measure between a current and a target pose. However, this requires to know robot kinematics and target position (through supervised visual tracking or measure). In a similar way, in Levine et al. (2015, 2016) (for tasks such as placing wooden rings or screwing bottle caps onto bottles), informative shaping rewards have been computed using a distance measure between current end-effector or manipulated object positions and their corresponding target positions. However, they require knowledge of kinematics or non-trivial visual modules. For similar tasks, a more sophisticated set-up has been proposed in Finn et al. (2016): the shaping reward is based on the distance between current visual features and target features, both of them being computed by an autoencoder. This requires to place the robot at the target position and extract target visual features each time the target location changes.

Another category of solutions to make learning with sparse-only rewards tractable consists in decomposing the whole problem into simpler sub-problems. For instance, assuming one goal state is known, a mechanism of learning from easy missions (Asada et al., 1996) can be used to learn very precise robotic manipulation tasks such as inserting and turning a key in a lock or assembling a gear onto an axle (Florensa et al., 2017). This method consists in starting learning the task from initial states close to the goal state and as far as learning improves, states are initialized further and further. Nevertheless, this method assumes the knowledge of a goal state and up to our knowledge, has not been proven efficient yet for a multi target position setting. Furthermore, hierarchical reinforcement learning can be used to decompose complex tasks such as block stacking into simpler sub-tasks. For instance, (Gudimella et al., 2017) quickly learns a block stacking task using Concept Network Reinforcement Learning (CNRL), a hierarchical framework which decomposes the problem into sub-problems like reaching the working area, grasping, reaching the second working area, and stacking. However, these sub-tasks use shaping rewards requiring kinematics and target pose knowledge. Besides, a similar task is learned using another hierarchical reinforcement learning framework called Scheduled Auxiliary Control (SAC-X) (Riedmiller et al., 2018). This uses auxiliary rewards (sparse for most of them) encouraging the robot to discover sub-goals such as making objects closer, making an object higher or lower than the other one, maximizing or minimizing the sum of finger tactile sensors. One key aspect of this architecture is that learning to achieve the sub-goals does not bias the learned policy and is only used to explore more the environment. However, some of these auxiliary rewards still require object tracking in the images and are not necessarily adaptable to any object.

In this paper, we consider the task of touching an object with the end-effector palm and we propose to learn it by decomposing the whole problem into simpler sub-problems and by using minimal prior knowledge. In other terms, our approach does not use kinematic models, hand-crafted features, calibration parameters and supervised visual modules. The task more precisely consists of reaching an object put on a table with the end-effector palm at several object positions and from several initial arm positions. This task can be considered and used as a pre-grasping task because target arm joint angles for our task are very close to target arm joint angles for grasping. The difficulty of our task relies on the fact that the arm has to reach from a large set of initial conditions (different object positions and initial arm positions, see **Figure 5**) so that it frequently has to substantially modify its orientation to reach the target with the palm. In this paper, we extend our prior work de La Bourdonnaye et al. (2018) to the more complex setting of multiple object positions. Besides, we conduct additional experiments to study of the influence of different reward terms. In this work, we have taken inspiration from the human development (Fischer, 1980; Carey et al., 1997) and developmental robotics (Hoffmann et al., 2005). To grasp an object, humans usually fixate it first, and then grasp it. This assertion does not mean that the only way to localize an object is to bring it in the fovea. Indeed, expert jugglers use information in the periphery of vision to detect juggling balls (Huys and Beek, 2002) and a monkey study reported that 81% of the neurons of the parietal reach region encode location in eye-centered coordinates (Batista et al., 1999). However, it can be a sufficient tool if these neurons are deficient (in case of widespread cortical atrophy Carey et al., 1997) and has the advantage of being compact. The rationale of our method is that an informative shaping reward for the object touching task can be constructed from the knowledge of simpler anterior tasks learned with minimal supervision. More precisely, the robot first learns to fixate objects (de La Bourdonnaye et al., 2017) and its own end-effector using a single deep reinforcement learning framework with little prior knowledge in the goal specification. Based on these two skills, an informative shaping reward is built, efficiently guiding the robot toward goal state areas. Our experiments show that learning this task with our weakly-supervised stage-wise framework yields same reaching performances as with a supervised reward, while learning with a sparse reward is slow. Our contribution is the design of our weakly-supervised framework which is efficient to learn to reach objects at several object positions and from several initial arm positions in a single shot.

The remainder is organized as follows. Section 2 presents basics about deep reinforcement learning, our stage-wise framework for reaching learning and the experimental protocol designed to validate our framework. Section 3 describes the results obtained and section 4 discusses the work from a broader perspective.

2. METHODS AND MATERIALS

This section presents the methods and the materials used in our experiments.

2.1. Background

Our work uses deep reinforcement learning. This section provides basics of reinforcement learning and the algorithm used to learn the different stages.

2.1.1. Reinforcement Learning

Reinforcement learning (RL) is a class of algorithms used to solve sequential decision making problems through learning. It is distinguishable from the dynamic programming category in that it does not require prior knowledge about dynamics and the reward signal. Most RL algorithms are based on Markov decision processes $< S, A, R, T >$ where S is the set of states, A the set of actions, T the transition model ($T : S \times A \to S$) and R the reward function ($R : S \times A \to \mathbb{R}$).

The source of learning comes from interaction between the agent and the environment and is composed of tuples $< \boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s'} >$ called transitions. $\boldsymbol{s}$ represents a state value and a the action performed at state $\boldsymbol{s}$. After the execution of the action $\boldsymbol{a}$, the agent receives a reward r and reaches a new state $\boldsymbol{s'}$.

The goal of an RL agent is to adapt its behavior to maximize a criterion linked to the future rewards. In the paper, we consider the sum of discounted future rewards as a learning goal: $J = \sum_{k=0}^{\infty} r_k \gamma^k$, where $\gamma \in [0, 1]$ is a discount factor and r_k the reward value at step k.

In our work, to optimize the criterion, we train a deterministic policy $\pi : S \to A$ jointly with the state-action value function Q in an actor-critic set-up:

$$Q_\pi(\boldsymbol{s}, \boldsymbol{a}) = E_\pi \left[\sum_{k=0}^{\infty} r_k \gamma^k \middle| \boldsymbol{s}, \boldsymbol{a} \right], \quad (\boldsymbol{s}, \boldsymbol{a}) \in S \times A. \tag{1}$$

2.1.2. Deep Reinforcement Learning

The curse of dimensionality (Bellman, 1961), the problem of representing RL functions with a large input space was explored with neural networks a long time ago (Tesauro, 1994). However, the use of neural networks for RL became more and more popular with the arrival of GPUs and the emergence of deep learning since high-dimensional state spaces could be used without requiring hand-crafted features. For instance, deep autoencoders were used to reduce the state space (composed of raw image pixels) of a Q function (Lange and Riedmiller, 2010) in an unsupervised way. Furthermore, deep convolutional neural networks were utilized to approximate the Q function (DQN: deep Q network) of an agent playing Atari games and outperforming human players (Mnih et al., 2015) directly from raw image pixels.

In our work, we use the DDPG algorithm (Lillicrap et al., 2016) which can solve RL problems with a high-dimensional state space and a continuous action space (like several other candidate algorithms). This algorithm combines the off-policy deterministic policy gradient algorithm (Silver et al., 2014) and the DQN.

DDPG is an "actor-critic" algorithm updating the critic Q_ϕ with parameters ϕ and the deterministic policy $\pi_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ as follows. At each time-step, we choose a mini-batch of N_b transitions from a large memory buffer of size N_trans using a uniform distribution:

$$< \boldsymbol{s_i}, \boldsymbol{a_i}, r_i, \boldsymbol{s'_i} >_{i \in \{1, \ldots, N_\text{b}\}} \in S \times A \times \mathbb{R} \times S.$$

The targets of the $Q_{\boldsymbol{\phi}}$ neural network are computed using a TD(0) update with a learning rate equal to 1:

$$\forall i \in \{1, \ldots, N_\text{b}\}, y_i = r_i + \gamma Q_{\phi'} \left(\boldsymbol{s'_i}, \boldsymbol{\pi_{\theta'}}(\boldsymbol{s'_i}) \right). \tag{2}$$

ϕ' and $\boldsymbol{\theta'}$ are the parameters of the target networks updated using a rate parameter τ (t denotes a time-step):

$$\phi'_{t+1} = \tau \phi_t + (1 - \tau) \phi'_t, \quad \boldsymbol{\theta'}_{t+1} = \tau \boldsymbol{\theta_t} + (1 - \tau) \boldsymbol{\theta'_t}, \tag{3}$$

The Q_ϕ network updates its weights by minimizing the squared error $\frac{1}{2N_\text{b}} \sum_{i=1}^{N_\text{b}} \left(y_i - Q_{\boldsymbol{\phi}}(\boldsymbol{s_i}, \boldsymbol{a_i}) \right)^2$. Using target networks greatly contributes to the learning stability of the neural networks and using a memory buffer helps to satisfy the constraint of i.i.d samples for learning with neural networks.

Using the $Q_{\boldsymbol{\phi}}$ network and the fact that the policy is deterministic, the following policy gradient is derived:

$$\frac{\partial Q_{\boldsymbol{\phi}}}{\partial \boldsymbol{\theta}} \simeq \frac{1}{N_b} \sum_{i=1}^{N_b} \frac{\partial Q_{\boldsymbol{\phi}} \left(\boldsymbol{s_i}, \boldsymbol{\pi_\theta}(\boldsymbol{s_i}) \right)}{\partial \boldsymbol{a}} \frac{\partial \boldsymbol{\pi_\theta}(\boldsymbol{s_i})}{\partial \boldsymbol{\theta}}. \tag{4}$$

This update makes the policy select the actions that maximize the Q function at the batch states. In addition to this algorithm, we use the inverting gradient procedure of Hausknecht and Stone (2016) to bound the actions. This method downscales the gradient when the action computed by the policy approaches its limit. When it exceeds its limit, the gradient is inverted. This mechanism prevents the actions from becoming too large.

2.2. Overview

We describe here the stage-wise learning process (see **Figures 1**, **2** for a schematic view). For our work, we use a 7 DOF arm with a pair of cameras as shown in **Figure 1**. The task consists of touching an object on a table with the end-effector palm of the robot. In the following, we use the notations:

- $\mathbf{I} = (\boldsymbol{I}^\text{left}, \boldsymbol{I}^\text{right})$ represents the images from the left and right cameras.
- $\mathbf{q} = (\boldsymbol{q}^\text{camera}, \boldsymbol{q}^\text{robot})$ represents the 3 camera joint angles (one common tilt angle and two independent pan angles) and 7 robot arm joint angles.
- $\boldsymbol{c}_\text{b}$ is a vector composed of 8 binary values associated with 8 areas of robot fingers. The 8 areas correspond to the proximal, medial and distal areas of the three fingers, with the exception of the proximal area of one finger which is linked to the palm. One binary value becomes 1 when its associated area is in contact with the object and 0 otherwise.

Our stage-wise learning framework is inspired by one of the human ways to locate an object: one can stare an object to locate

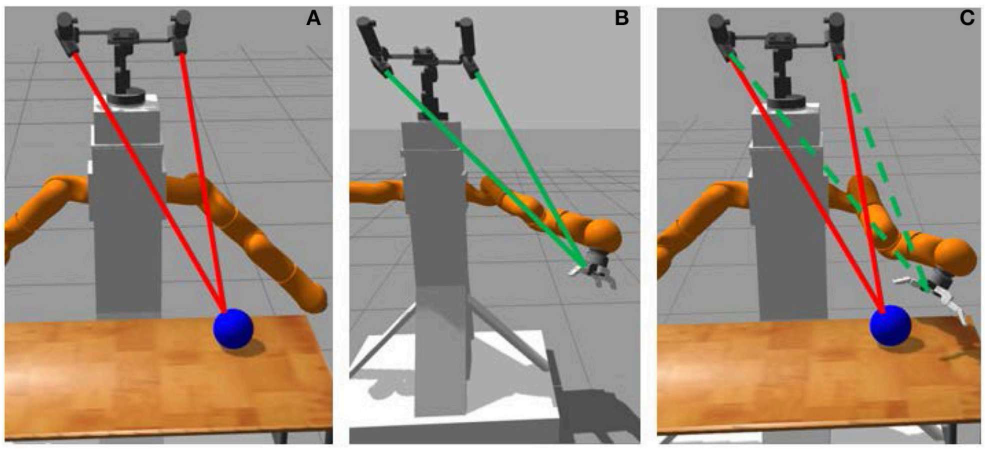


FIGURE 1 | Palm-touching learning process: **(A)** object fixation, **(B)** end-effector fixation and hand-eye coordination, **(C)** palm-touching.

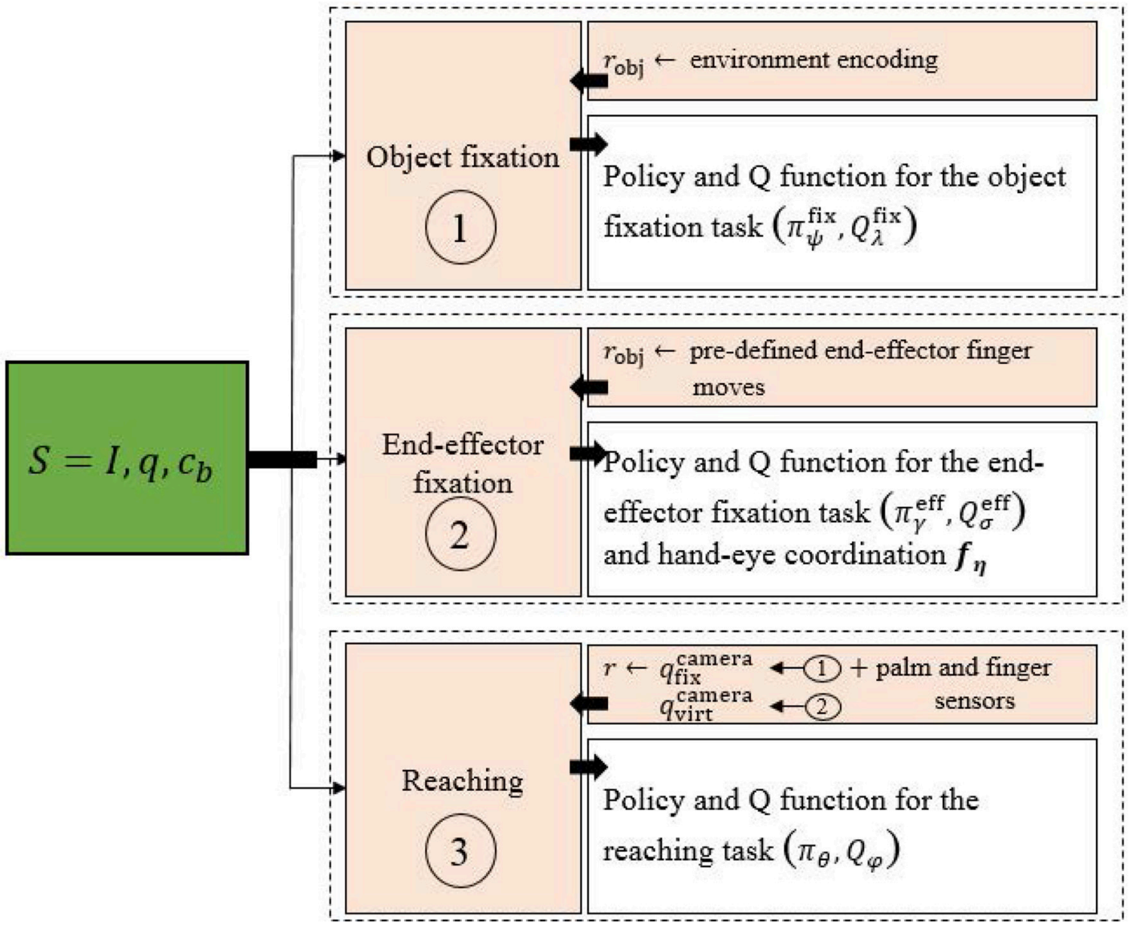

$S = I, q, c_b$
Object fixation
1
$r_{obj} \leftarrow$ environment encoding
Policy and Q function for the object fixation task $(\pi_\psi^{fix}, Q_\lambda^{fix})$
End-effector fixation
2
$r_{obj} \leftarrow$ pre-defined end-effector finger moves
Policy and Q function for the end-effector fixation task $(\pi_\gamma^{eff}, Q_\sigma^{eff})$ and hand-eye coordination f_η
Reaching
3
$r \leftarrow q_{fix}^{camera}$ ← 1 + palm and finger sensors
q_{virt}^{camera} ← 2
Policy and Q function for the reaching task (π_θ, Q_φ)

FIGURE 2 | Overall scheme of the touching task learning procedure. Greek subscripts represent neural network parameters.

it. The main objective is to apply this principle with minimal supervision. The proposed method involves three successive tasks:

First, the robot learns from raw pixels to fixate the object with a two-camera system. For this, we use (de La Bourdonnaye et al., 2017) to learn to fixate an object with weak supervision.

At the end of the fixation, the camera system coordinates $\boldsymbol{q}_{\text{fix}}^{\text{camera}}$ implicitly encode the object position in 3D space.

Second, the robot learns a hand-eye coordination function f_η which maps robot joint coordinates to virtual camera coordinates:

$$\boldsymbol{q}_{\text{virt}}^{\text{camera}} = f_\eta(\boldsymbol{q}^{\text{robot}}). \tag{5}$$

These virtual camera coordinates correspond to the camera coordinates which would make the camera system look at the end-effector. Finally, a reward signal using $\boldsymbol{q}_{\text{fix}}^{\text{camera}}$ and $\boldsymbol{q}_{\text{virt}}^{\text{camera}}$ to make the end-effector close to the object is computed. It is combined with a sparse reward, indicating if the end-effector palm touches the object or not and a term penalizing end-effector contacts with the table (which assumes that the robot has the touching ability to distinguish the object from the table). In the following, we describe each of the three steps.

2.3. Learning Binocular Object Fixations

In this part we describe the object fixation learning which the first stage of our method.

2.3.1. Task Overview

We define object fixation as bringing the object at the center of $\boldsymbol{I}_{\text{left}}$ and $\boldsymbol{I}_{\text{right}}$ by moving the cameras. To learn it, we build on our prior work (de La Bourdonnaye et al., 2017), which we summarize below for sake of clarity.

The task is learned with the DDPG algorithm (Lillicrap et al., 2016) using $\boldsymbol{I}$ and $\boldsymbol{q}_{\text{camera}}$ as states, and $\boldsymbol{\Delta q}_{\text{camera}}$ as actions. The reward function is the sum of left and right camera components: $r_{\text{obj}} = r_{\text{obj}}^{\text{left}} + r_{\text{obj}}^{\text{right}}$. For each camera cam = left or right, the reward function r_{obj}^{cam} is an affine decreasing function of the distance between the image center x_c and the estimated object position x_{obj}^{cam}:

$$r_{\text{obj}}^{cam} = 2\frac{\frac{1}{2}d_{\max} - ||x_c - x_{\text{obj}}^{cam}||_2}{d_{\max}} \in [-1, 1], \tag{6}$$

with $d_{\max}$ being the maximal distance between the image center and the object pixellic position.

An episodic set-up is used. For each episode, a random object is put at a random location above the table. The episode ends when a given number of transitions ($N_e = 35$) has been reached. Section 2.3.2 describes how x_{obj} is obtained with minimal supervision.

2.3.2. Object Detection

The object detection mechanism involves a convolutional autoencoder training step in which images of the environment without object are encoded. To do this, we use two 10,000-sized databases (one for each camera) of pictures captured without object in the environment. The camera positions cover a regular grid between joint limits which are set on purpose to keep the table inside the field of view. After the image acquisition, two autoencoders $A_{\boldsymbol{\mu}^{\text{left}}}$ and $A_{\boldsymbol{\mu}^{\text{right}}}$ are trained on each database using ADAM (Kingma and Ba, 2015). Before training, the images are converted for computational purposes from RGB 200 × 200 to 50 × 50 grayscale images.

When the robot is learning to fixate objects, we assume that the object is in the environment. It is detected as an anomaly and localized in the images $\boldsymbol{I}^{\text{left}}$ and $\boldsymbol{I}^{\text{right}}$. Indeed, we assume objects are badly reconstructed because they are not present in the database images for autoencoder training so that the reconstruction error intensity is higher at the object position. The steps of object detection are presented in **Figure 3**.

After grayscale conversion and downsampling steps, the autoencoder reconstruction error images $|\boldsymbol{I}^{\text{left}} - \hat{\boldsymbol{I}}^{\text{left}}|$ and $|\boldsymbol{I}^{\text{right}} - \hat{\boldsymbol{I}}^{\text{right}}|$ are computed ($\hat{\boldsymbol{I}}^{\text{cam}} = A_{\boldsymbol{\mu}^{cam}}(I^{cam})$). From these error maps, we extract the N points $\{\boldsymbol{x(i)}\}_{i\in\{1,\ldots,N\}}$ which have the highest intensity. $\{L(i)\}_{i\in\{1,\ldots,N\}}$ is the set of corresponding luminances. Then, we compute a discrete probability distribution using a kernel density estimator with a Gaussian kernel of zero mean and unit variance:

$$\forall i \in \{1, \ldots, N\}, p(i) = \frac{1}{N}\sum_{j=1}^{N} L(j)K(\boldsymbol{x_i} - \boldsymbol{x_j}), \tag{7}$$

with $K(\boldsymbol{x_i} - \boldsymbol{x_j}) = \frac{1}{2\pi}\exp^{-0.5||\boldsymbol{x_i}-\boldsymbol{x_j}||_2^2}$.

After that, the estimated object pixellic positions, respectively $x_{\text{obj}}^{\text{left}}$ and $x_{\text{obj}}^{\text{right}}$ are at the maximal probabilities:

$$x_{\text{obj}}^{cam} = x_{\arg\max_i(p(i))} \tag{8}$$

This object detection principle only requires an autoencoder pre-training step without object and the assumption that there is an object in the scene subsequently. Note that the potential noise of this object detection has been tackled using a learning method in de La Bourdonnaye et al. (2017).

2.4. Learning a Hand-Eye Coordination Function f_η

We now describe how a similar framework can be used to learn end-effector fixation and a robot hand-eye coordination function.

2.4.1. Task Overview

We model the hand-eye coordination function f_η (see Equation 5) with a neural network. To learn it, we need to have a database D of input-output pairs ($\boldsymbol{q}^{\text{robot}}, \boldsymbol{q}^{\text{camera}}$) where $\boldsymbol{q}^{\text{camera}}$ makes the camera look at the end-effector. To produce such samples, we learn to fixate the end-effector. For this, we use a similar framework as the object fixation task. We use the DDPG algorithm and a reward requiring weak supervision. The Markov Decision Process is the same as for the object fixation with the exception of the reward function. The latter involves the end-effector detection x_{eff}^{cam} instead of x_{obj}^{cam}:

$$r_{\text{eff}}^{cam} = 2\frac{\frac{1}{2}d_{\max} - ||\boldsymbol{x_c} - x_{\text{eff}}^{cam}||_2}{d_{\max}} \in [-1, 1]. \tag{9}$$

The set-up is also episodic. For each episode, random arm joint coordinates are generated using uniform distributions with fixed limits. They are empirically set to provide a large variety of reachable arm configurations.

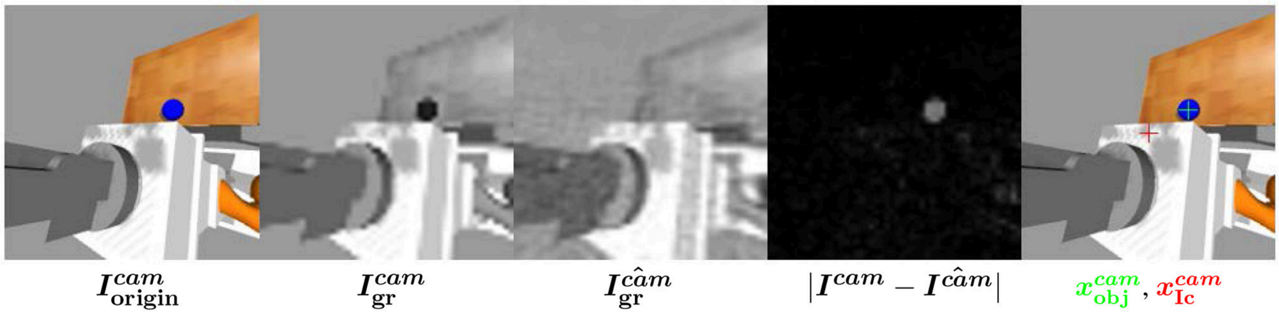


FIGURE 3 | Object detection computation scheme (de La Bourdonnaye et al., 2017).

During learning, training pairs $(\boldsymbol{q}^{\text{robot}}, \boldsymbol{q}^{\text{camera}})$ are added to D when the reward r^{cam}_{eff} is above a fixed threshold. When the number of samples in D is higher than the batch size N_{bf}, we train f_η on random batches of D each time a new sample is added to D.

2.4.2. End-Effector Detection

We describe here how we detect the end-effector in the image. Unlike de La Bourdonnaye et al. (2017) which uses an autoencoder to localize the object, the end-effector image position is computed using the difference in the image before and after pre-defined end-effector finger moves (Metta and Fitzpatrick, 2003). The idea is that the hand is segmented from the rest of the scene because its appearance varies according to finger moves. Then, this end-effector detection method only requires to specify finger moves.

Figure 4 presents the different steps of the end-effector detection:

- The images before (I^{cam}_{before}) and after (I^{cam}_{after}) the end-effector moves are saved.
- The difference of images is calculated and the end-effector position x^{cam}_{eff} is computed using a kernel density estimator the same way x^{cam}_{obj} is calculated from the autoencoder reconstruction error image.

Note that the end-effector detection is also filtered as in section 2.3.2.

2.5. Learning to Touch

In this section, we describe how the previous learned tasks help to learn to touch the object.

2.5.1. Task Overview

The touching task consists of reaching with the end-effector palm the object above the table at different reachable positions from a large set of initial arm positions (see **Figure 5** for a display of 8 randomly generated initial configurations). The goal of learning to reach both at different target positions and from a large set of initial robot joint angles is mainly motivated by the fact it allows to learn policies that are more robust to perturbations in the joint space (Rajeswaran et al., 2017). In addition, reaching from different initial joint angles allows to reach from positions with a badly oriented end-effector which is a challenging task.

The objective of the task is defined by a sparse reward term r_{sparse} which indicates if there is palm-touching or not:

$$r_{\text{sparse}} = \begin{cases} 1, & \text{if success,} \\ p_{\text{time}} \in \mathbb{R}^-, & \text{otherwise.} \end{cases} \tag{10}$$

Note that the negative term p_{time} ensures that the robot looks for the quickest path to the goal. The state space S is composed of the arm and camera joint angles $\boldsymbol{q}$ as well as eight binary tactile sensors $\boldsymbol{c}_{\text{b}}$ attached to the fingers of the Barrett Hand. Images are not required here because we use a single object and consider that the camera joint angles give sufficient information about the 3D object position. However, they would be necessary if objects with different shapes were used in the experiments. The actions are variations of the robot joint angles: $\boldsymbol{a} = \boldsymbol{\Delta q}^{\text{robot}}$ which are seven real-valued scalars.

2.5.2. Reward Computation

To compute the touching reward function, we use the object binocular fixation policy and the hand-eye coordination function. After the execution of an object fixation step (using the object fixation policy $\boldsymbol{\pi_\psi}$), we get the fixation camera angles $\boldsymbol{q}^{\text{camera}}_{\text{fix}}$ which implicitly encode the object 3D position. After that, using Equation (5) at each time-step, the hand-eye coordination function f_η gives us $\boldsymbol{q}^{\text{camera}}_{\text{virt}}$ which implicitly encodes the end-effector 3D position. Then, a reward shaping term r_{shCam} can be computed:

$$r_{\text{shCam}} = \begin{cases} 0, & \text{if success,} \\ c_{\text{cam}}||\boldsymbol{q}^{\text{camera}}_{\text{fix}} - \boldsymbol{q}^{\text{camera}}_{\text{virt}}||_2 - p_{\text{time}}, & \text{otherwise.} \end{cases} \tag{11}$$

with $c_{\text{cam}} \in \mathbb{R}^-$. r_{shCam} represents an informative term which depends on the distance between the virtual camera coordinates and the camera coordinates which make the camera system fixate the object. Thus, it encourages the end-effector to be close to the object. Note that the slope c_{cam} is chosen to ensure shaping rewards are small compared with the non-zero sparse reward.

Using these sole terms yields decent performances but we observed that the robot was badly guided when it is close

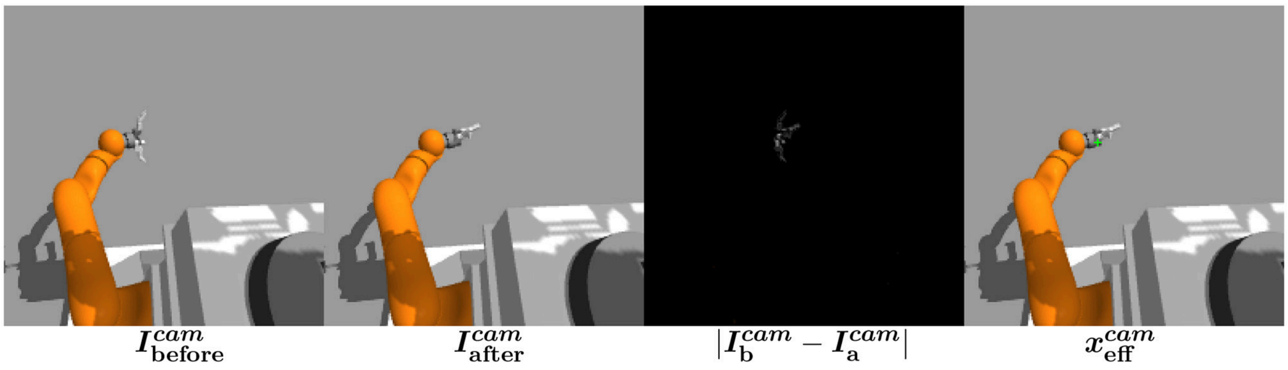

I^{cam}_{before}
I^{cam}_{after}
$|I^{cam}_{\text{b}} - I^{cam}_{\text{a}}|$
x^{cam}_{eff}

FIGURE 4 | End-effector detection computation scheme.

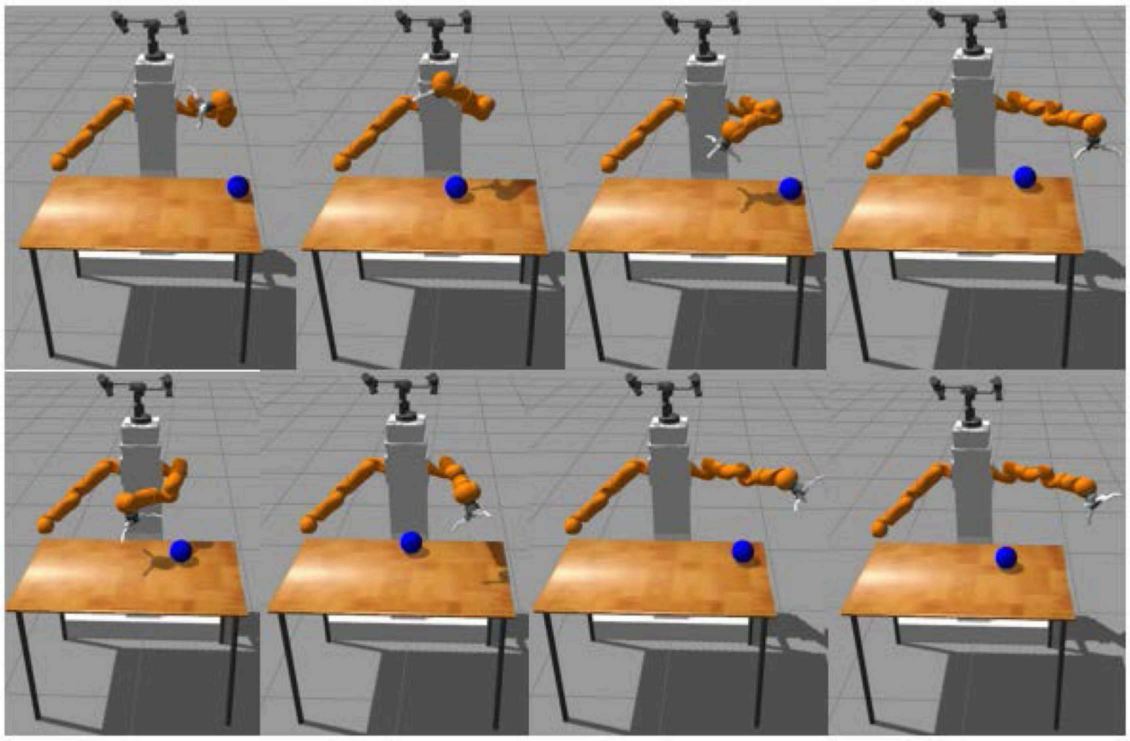


FIGURE 5 | Random examples of initial configurations.

to the table. Indeed, fewer moves are physically plausible and the algorithm can take time to learn them. To accelerate this selection, we propose a new tactile reward term $r_{\text{penContact}}$ to the reward function penalizing states where the end-effector is in contact with the table:

$$r_{\text{penContact}} = \begin{cases} p_{\text{contact}} \in \mathbb{R}^-, & \text{if contact between the end-effector and the table,} \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

Indeed, by applying penalties, we hope that the robot explores areas where it is not in contact with the table, i.e., where it can move without too many constraints to the goal. To compute this term, we make the assumption that the robot knows from its tactile sensors whether it is touching the table.

Finally, the reward function is built from the three previous terms:

$$r_{\text{proposedPen}} = r_{\text{sparse}} + r_{\text{penContact}} + r_{\text{shCam}} \quad (13)$$

The relative effect of each of these terms is evaluated in the experiments.

2.6. Experiments

In this section, we describe the experimental protocol. The objective of experiments is to evaluate learning performances using the proposed reward function and other ones because they allow to evaluate the relative impacts of each reward term. Besides, we wish to evaluate whether our weakly supervised reward can reach same performances as with a supervised counterpart.

2.6.1. Different Reward Functions

The reward functions which will be used in our experiments are listed below:

- $r_{\text{sparse}} = \begin{cases} 1, & \text{if success,} \\ p_{\text{time}}, & \text{otherwise.} \end{cases}$
  This reward is described by Equation (10) and rewards the robot only when the palm touches the object. Besides, it penalizes each unsuccessful movement to encourage the robot to quickly touch the object. Note that using such a sparse reward means that we only dispense with the hand-eye coordination information. Indeed, information brought by the object fixation (the camera joint angles) is still present in the state space.
- $r_{\text{proposedPen}} = r_{\text{sparse}} + r_{\text{penContact}} + r_{\text{shCam}}$
  This is the proposed reward function (described in Equation 13).
- $r_{\text{proposed}} = r_{\text{sparse}} + r_{\text{shCam}}$
  This is the proposed reward function without the penalties for the contact between the end-effector and the table. This is used to show the influence of the penalty in the learning procedure.
- $r_{\text{sparsePen}} = r_{\text{sparse}} + r_{\text{penContact}}$
  We add to r_{sparse} a term penalizing contacts of the end-effector with the table.
- $r_{\text{supervisedPen}} = r_{\text{sparse}} + r_{\text{penContact}} + \begin{cases} 0, & \text{if success,} \\ c_{\text{cart}}||\boldsymbol{p} - \boldsymbol{p}_{\text{target}}||_2 + 0.0125, & \text{otherwise,} \end{cases}$
  with $c_{\text{cart}} < 0$. To build this reward, we give a 3-dimensional end-effector target Cartesian pose $\boldsymbol{p}_{\text{target}}$ for the shaping part and we add a sparse reward as well as a term penalizing end-effector contacts with the table. This reward is the closest to the proposed $r_{\text{proposedPen}}$ but its shaping term requires forward kinematics and 3D object pose information. Finally, the slope c_{cart} is chosen to make the shaping term take about the same values as r_{shCam}.

Note that we choose not to compare our reward function with a Cartesian shaping reward without a sparse term. Indeed, for such a reward function, a success would be to touch with the palm from a specific orientation and position. In our case, a success can be to touch with the palm in any position. The tasks are then too different to be compared in terms of touching improvement.

2.6.2. Material

We describe here the material we use for our experiments.

Figure 6 presents the chosen virtual experimental platform which is the realistic representation of one of our real robotic platforms. The simulations use the Gazebo simulator with the ROS (Quigley et al., 2009) middleware. The robotic platform is composed of three entities:

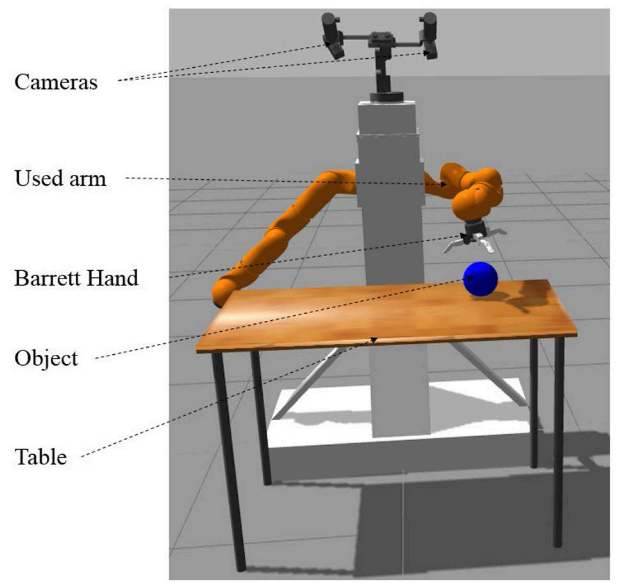


FIGURE 6 | Scheme of the robotic platform.

- A two-camera pan-tilt system attached above the platform
- Two robotic arms attached on the left and right sides of the platform. Note that we use only one arm in our experiments.
- A Barrett hand is attached to the arm that we use in the experiments.

A table from the Gazebo object database is placed below the cameras and in front of the bi-arm platform. This table is not present when we learn the hand-eye coordination function. To learn object fixation (de La Bourdonnaye et al., 2017), we use some objects from the gazebo object database and several hand-made ones with various shapes and colors (see **Figure 7**). We use a blue-ball for the reaching experiments though the method does not depend on this specific model since the robot learns to look at any object.

2.6.3. Experimental Protocol

We describe how we compare the policies learned with different reward signals for the touching task. The protocol contains a training and a test phase.

2.6.3.1. Training phase

For training, we use the DDPG algorithm (Lillicrap et al., 2016) and the previously defined reward functions. Learning happens on N_{tot} bounded-length episodes of maximal size $N_{\max}$. Each episode has an initial arm position and an object position. The object position is uniformly chosen from a rectangular area of reachable object positions on the table. The initial robot joint angles are sampled from a set of uniform distributions (each one corresponding to a robot joint angle). When an initial position

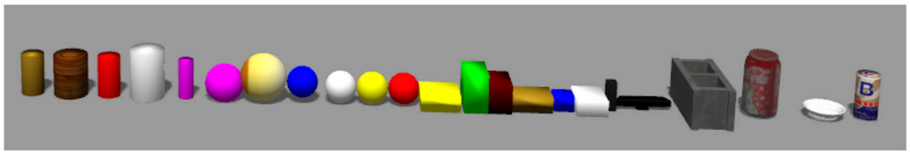


FIGURE 7 | Training set for the object fixation task (de La Bourdonnaye et al., 2017).

leads to a collision between the arm and its environment, the initial position is re-set until a collision-free position is sampled.

For the exploration, we use the Ornstein-Uhlenbeck process. This correlates the noise $\epsilon_j(t)$ for a joint at time t with the noise $\epsilon_j(t-1)$ of the same joint at time $t-1$ with the equation:

$$\epsilon_j(t) = \theta_j\mu_j + (1-\theta_j)\epsilon_j(t-1) + \left(\xi_j(t) \sim \mathcal{N}\left(0, \sigma_j\right)\right). \quad (14)$$

θ_j is a factor trading-off the correlation with the previous noise and the correlation with the equilibrium value μ_j and σ_j is the standard deviation of the used Gaussian distribution. This exploration procedure is particularly interesting in problems in which the same action applied during several time-steps can be the optimal behavior.

As the task requires a precise orientation of the end-effector, the robot frequently blocks itself close to a reaching position. For instance, the robot can touch the object with its fingers without touching it with the palm. And, if the actions computed by the policy make the end-effector move downward, the robot can be blocked by the table despite exploration. Thus, to avoid these situations, we handle the times when the robot is blocked without succeeding in reaching. More precisely, when the robot is blocked a backward action is taken, i.e., the robot goes back to a previous contact-less position. This allows to more correctly discriminate actions in the contact areas in the sense the robot is provided with other chances of success.

Through the training experiments, we wish to compare our reward requiring little supervision with other ones. Consequently, for all the reward signals, in order to monitor the learning progress, we specifically plot the reaching frequency ν^{reward} over the episodes, reward referring to a specific reward function. We average six experiments per setting and provide confidence interval plots $[m^{\text{reward}}, M^{\text{reward}}]$ for each computed average. m^{reward} and M^{reward} are computed according to the equations below:

$$m^{\text{reward}} = \nu^{\text{reward}} - \frac{1.96\sigma^{\text{reward}}}{\sqrt{N_{\text{run}}}}, \quad (15)$$

$$M^{\text{reward}} = \nu^{\text{reward}} + \frac{1.96\sigma^{\text{reward}}}{\sqrt{N_{\text{run}}}}, \quad (16)$$

with N_{run} being the number of runs per reward function and σ^{reward} the standard deviation of the reaching frequency for each reward function.

Furthermore, we record N_1 the number of episodes it took to reach a first reaching success, N_{90} the number of RL iterations it took to reach and remain above reaching performance of 90 % as well as associated confidence intervals and standard deviations. These variables are used to evaluate the learning velocity with different reward functions.

2.6.3.2. Test phase

To evaluate the learned policies, we apply them without any exploration noise on N_{tot} random episodes and we compute the touching frequency $\nu_{\text{test}}^{\text{reward}}$. Moreover, for each reward signal, we provide a confidence interval for the average touching frequency and the standard deviation of the touching frequency $\sigma^{\nu_{\text{test}}^{\text{reward}}}$. Note that we do not apply the systematic backward motion used in the training phase to deal with blocked situations. Instead, when the robot is blocked, it just follows the learned policy. Like in the training phase, the results are averaged over six experiments per setting.

2.6.4. Implementation Details

For all the neural network algorithms, we use the caffe library (Jia et al., 2014). A GPU (nvidia GeForce GTX Titan X) is used for the experiments.

We use the same neural network architectures as in de La Bourdonnaye et al. (2017) for the end-effector and object fixation tasks. The hyperparameter values are also the same with the exception of the number of iterations: 200, 000. The hand-eye coordination function is a neural network with 2 fully connected hidden layers of 10 and 5 neurons and a batch size N_{bf} of 32 is used to learn it. For the episode initialization of the end-effector fixation task, the seven arm joint angle distribution amplitudes are 11, 46, 69, 92, 92, 86, and 0° if we consider the ascending order in the kinematic chain i.e., from the base link to the end-effector.

For the touching task, the Q network has 3 fully connected layers with 250, 200, and 1 neural units. The policy network involves 3 fully connected layers with 200, 150, and 7 neural units. The weights are updated using the Adam solver (Kingma and Ba, 2015). **Tables 1**, **2** provide values for the parameters used in the experiments. For the Q update, the discount factor γ is equal to 0.99. For the episode initialization, the distribution limits are 23, 57, 80, 91, 103, 80, and 11°.

3. RESULTS

Table 3 presents the final performances of the different policies as well as the number of episodes it takes to get a first reaching success and the number of RL iterations it takes to reach (and remain above) an average reaching performance of 90 %. The average reaching performance is obtained using exponential

TABLE 1 | Parameter values.

Parameters	N_{max}	N_b	c_{cam}	c_{cart}	N_{trans}	N_{tot} (training)	N_{tot} (test)	$p_{contact}$	p_{time}
Values	100	256	$-\frac{1}{30}$	$-\frac{1}{40}$	60,000	40,000	1,000	−0.01	−0.0125

TABLE 2 | Ornstein-Uhlenbeck process parameters.

Parameters	$\theta_j, j \in \{1,\dots,7\}$	$\mu_j, j \in \{1,\dots,7\}$	$\sigma_j, j \in \{1,\dots,4\}$	$\sigma_j, j \in \{5,\dots,7\}$
Values	0.8	0	0.01	0.04

TABLE 3 | Values featuring learning velocity (N_1 and N_{90}), final reaching frequency (ν_{test}^{reward}) and associated standard deviations.

Reward	$r_{proposedPen}$	$r_{proposed}$	$r_{supervisedPen}$	$r_{sparsePen}$	r_{sparse}
N_1	95 ± 16	110 ± 19	105 ± 29	7505 ± 6918	8214 ± 4145
N_{90}	(1.73 ± 0.25) × 10^6	(2.35 ± 0.17) × 10^6	(1.55 ± 0.13) × 10^6	N/A	N/A
ν_{test}^{reward} (%)	94.9 ± 1.3	90.7± 2.83	97.2 ± 0.67	59.9 ± 37.6	81.9 ± 17.3
σ^{N_1}	20	23	36	8646	5180
$\sigma^{N_{90}}$	312,911	172,129	165,160	N/A	N/A
$\sigma^{\nu_{test}^{reward}}$ (%)	1.64	3.53	0.84	47	21.6

smoothing: $\nu_f^{reward} = (1-\omega)\nu_f^{reward} + \omega\nu_r^{reward}$, with ν_r^{reward} and ν_f^{reward} being the raw and smoothed frequencies and ω the smoothing factor being equal to 0.003. **Figure 8** shows the experimental training curves as well as associated confidence intervals. Note that this figure use exponential smoothing for visualization purposes. We can notice several important facts:

- Learning the reaching task can work very well because the robot reaches 90% of touching performances with the reward functions using shaping terms (Videos of the policy learned with our reward function can be consulted in the **Supplementary Material**). This shows that the camera joint angles integrated in the state space encode sufficiently well the object position, which confirms the rationale of our method.
- With the use of sparse-only rewards, the probability of getting the first success is low. It takes a lot of episodes to reach a first success (from 7,505 episodes for the N_1 values). Furthermore, we cannot have a precise idea about the time when the first success occurs because the standard deviations are very high. Moreover, N_{90} values are not available for these two reward settings because some of the runs were not successful at all. Finally, as shown by **Figure 8**, the confidence intervals for the average reaching frequency are very large, which means that the average estimation is not precise at all for the sparse reward settings. The only fact we can notice for these settings is that it can work for a run and totally fails for another one. Then, these reward functions do not ensure a reliable learning.
- With a shaping term, the probability of having first successes is much higher. The different N_1 values for $r_{proposedPen}$, $r_{proposed}$, $r_{supervisedPen}$ are of the same order of magnitude, are small, and exhibit low standard deviations. And our weakly-supervised setting allows to approach similar reaching

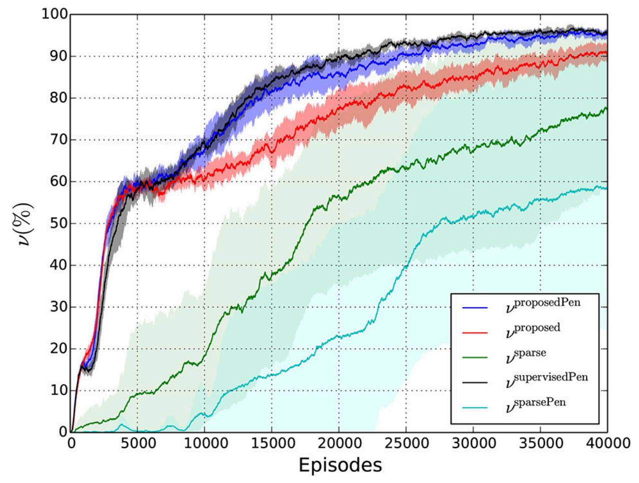


FIGURE 8 | Evolution of the average reaching frequency during training for the different reward functions.

performances compared with its supervised counterpart even if the final reaching frequency is slightly lower. In addition, **Figure 8** shows that the confidence intervals of $\nu^{supervisedPen}$ and $\nu^{proposedPen}$ intertwine even if the bounds of $\nu^{supervisedPen}$ are generally higher. This shows that even if $\nu^{supervisedPen}$ is higher than $\nu^{proposedPen}$ most of the time, results are close. Furthermore, we notice that three phases can be distinguished. From 0 to about 5,000 episodes, the reward curves increase with the same velocity. It corresponds to a phase in which some initial positions are mastered without substantial end-effector orientation changes. Indeed, for some initial positions, the robot has to change only a little its end-effector orientation to reach a grasping posture. After 5,000 episodes, there is a period of slow increase for the three settings and from about 7,000 episodes, "harder" initial positions are more and more mastered. We observe that the three settings start to distinguish from each other and the term penalizing contacts seems to be a decisive factor.

- Indeed, with a term penalizing contacts between the end-effector and the table, learning becomes faster. To show this, we can compare $r_{\text{proposedPen}}$ and r_{proposed}: N_{90} is lower for $r_{\text{proposedPen}}$, $\nu_{\text{test}}^{\text{proposedPen}}$ is higher than $\nu_{\text{test}}^{\text{proposed}}$, and **Figure 8** shows that $\nu^{\text{proposedPen}}$ is always superior to ν^{proposed} after 10,000 episodes. Furthermore, in **Figure 8** we notice that the upper bound M^{proposed} is generally inferior to the lower bound $m^{\text{proposedPen}}$. All of these observations show the supremacy of $r^{\text{proposedPen}}$ over r^{proposed}. It shows that penalizing contacts between the end-effector and the table has an important influence on learning performances. The reason is that it is easier to experiment "good" moves in contact-less areas given the robot can easily be blocked when it touches the table.

4. DISCUSSION

4.1. Contributions

Our first contribution is the design of a stage-wise learning framework to learn a complex reaching task. This framework involves the DDPG algorithm though any deep RL algorithm suitable to continuous action spaces could be used. Interestingly the first two tasks are largely similar in their modeling: we use the same MDPs with the exception of the reward function, the same kernel density estimator for localizing the point of interest in the image and the same filtering method to remove the detection noise. The knowledge of the two tasks are then combined to compute an informative shaping reward efficiently guiding the robot toward reaching postures.

Our second contribution is to learn the task with only weak supervision, i.e., without kinematics, calibration or pre-processing blocks and to exhibit similar performances compared with a fully supervised reward function. Furthermore, our framework is applied on a challenging task with a large set of initial configurations: several initial arm positions and several object positions as shown in **Figure 5**.

4.2. Related Work

Our approach resembles some developmental robotics methods which learn to reach using a hand-eye coordination function and object fixation. However, they are usually paired with supervision for the object and/or end-effector fixations (Nori et al., 2007; Chinellato et al., 2011; Jamone et al., 2012; Law et al., 2014) or computation of action primitives (Hoffmann et al., 2005). In other terms, object or end-effector detection use markers or simple blob-detection algorithms which would not be valid for any kind of object. The contributions brought by these papers are more related to the learning architecture which is close to the one of infants whereas our work focuses on reducing the amount of external information used for learning.

The multi-target-position multi-initial-arm-position setting has also been implemented on a simulated reaching task using a 7 DOF manipulator (Lillicrap et al., 2016). However, there were neither collision aspects nor orientation constraints for the end-effector and a supervised shaping reward was used. Lampe and Riedmiller (2013) learns an object grasping policy but integrates the object position in a camera image in a relatively low-dimensional state space, which requires a supervised visual module. Popov et al. (2017) learns a brick grasping task from several initial arm positions at several target positions. However, for the arm initialization, the end-effector is always made close to the object and its orientation adapted to a grasping action.

4.3. Limitations

Our approach has certain limitations mainly related to the first stages of the stage-wise framework. In the object fixation step, even though learning is weakly-supervised, if the environment varies, the approach in its current form needs the intervention of a human user to learn again to encode the environment. Besides, our approach constrains objects not to be present in the scene when the environment is encoded. And finally, the fixation cannot be applied on the object when the arm is above the table. Concerning the hand-eye coordination learning stage, the method implemented here requires an immobile background to make the end-effector detection method work. Note that this problem is solved in the literature by correlating finger moves with detection changes in the image Metta and Fitzpatrick (2003).

5. FURTHER RESEARCH

As further research, we wish to make the first and the second stages of our framework robust respectively to environment variations and moves in the background and also to be able to fixate the object when the arm is above the table. A good hint for this would be to achieve an open-ended learning framework in which the learnings of the tasks presented in the paper overlap and drive each other. For example, learning to reach an object with the end-effector may first help the robot to acquire the knowledge of what is an object and would consequently drive the learning of object fixation. Second, it may help the robot to acquire hand-eye coordination.

Furthermore, it would be interesting to learn other kinds of manipulation tasks including complex ones such as inserting a key in a lock with our framework. In principle, switching from a task to another one would just require to switch from a sparse reward to another one. However, some sparse rewards are less likely to be reached that other ones, e.g., grasping is less likely than palm-reaching. Consequently, our learning framework might not be directly applicable for too complex tasks, and learning them with our framework would be achieved by learning a curriculum of tasks, from the simplest to the most complicated. Finally, we wish to adapt the framework to a real robotic setting.

AUTHOR CONTRIBUTIONS

FdLB implemented the code and wrote the original version of the paper. JT, TC, and CT gave advice on the work itself and reviewed the paper.

ACKNOWLEDGMENTS

This work is sponsored by the French government research program Investissements d'avenir through the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01), by the European Union through the program Regional competitiveness and employment (ERDF Auvergne region), and by the Auvergne region. JT acknowledges support from the Quandt foundation and the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 713010 (GOAL-Robots Goal-based Open-ended Autonomous Learning Robots).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frobt.2018.00110/full#supplementary-material

Video S1 | Video showing the different learned behaviors from object fixation to reaching.

REFERENCES

Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K. (1996). Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learn*. 22, 279–303.

Batista, A. P., Buneo, C. A., and Andersen, R. A. (1999). Reach lans in eye-centered coordinates. *Science*. 285, 257–260.

Bellman, R. E. (1961). *Adaptive Control Processes: A Guided Tour*. Cambridge, MA: MIT Press.

Carey, D. P., Coleman, R. J., and Della Salla, S. (1997). Magnetic misreaching. *Cortex* 33, 639–652.

Chebotar, Y., Hausman, K., Zhang, M., Sukhatme, G. S., Schaal, S., and Levine, S. (2017). "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *ICML* (Sydney, NSW).

Chinellato, E., Antonelli, M., Grzyb, B. J., and del Pobil, A. P. (2011). Implicit sensorimotor mapping of the peripersonal space by gazing and reaching. *IEEE Trans. Auton. Ment. Dev*. 3, 43–53.

de La Bourdonnaye, F., Teulière, C., Triesch, J., and Chateau, T. (2017). "Learning of binocular fixations using anomaly detection with deep reinforcement learning," in *IJCNN* (Anchorage, AK).

de La Bourdonnaye, F., Teulière, C., Triesch, J., and Chateau, T. (2018). "Learning to touch objects through stage-wise deep reinforcement learning," in *IROS* (Madrid).

Deisenroth, M. P., Rasmussen, C. E., and Fox, D. (2011). "Learning to control a low-cost manipulator using data-efficient reinforcement learning," in *Robotics: Science and Systems* (Sydney, NSW).

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. (2016). "Deep spatial autoencoders for visuomotor learning," in *ICRA* (Stockholm).

Fischer, K. (1980). A theory of cognitive development: the control and construction of hierarchies of skills. *Psychol. Rev*. 87, 477–531.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. (2017). "Reverse curriculum generation for reinforcement learning," in *CoRL* (Moutain View, CA).

Ghadirzadeh, A., Maki, A., Kragic, D., and Björkman, M. (2017). "Deep predictive policy training using reinforcement learning," in *IROS* (Vancouver, BC).

Gu, S., Holly, E., Lillicrap, T. P., and Levine, S. (2017). "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *ICRA* (Singapore).

Gudimella, A., Story, R., Shaker, M., Kong, R., Brown, M., Shnayder, V., and Campos, M. (2017). Deep reinforcement learning for dexterous manipulation with concept networks. *CoRR*. Available online at: https://arxiv.org/abs/1709.06977 (Accessed Jun 25, 2018).

Hausknecht, M. J., and Stone, P. (2016). "Deep reinforcement learning in parameterized action space," in *ICLR* (San Juan).

Hoffmann, H., Schenck, W., and Möller, R. (2005). Learning visuomotor transformations for gaze-control and grasping. *Biol. Cybern*. 93, 119–130. doi: 10.1007/s00422-005-0575-x

Huys, R., and Beek, P. J. (2002). The coupling between point-of-gaze and ballmovements in three-ball cascade juggling: the effects of expertise, pattern and tempo. *J. Sports Sci*. 20, 171-186. doi: 10.1080/026404102317284745

Jamone, L., Natale, L., Nori, F., Metta, G., and Sandini, G. (2012). Autonomous online learning of reaching behavior in a humanoid Robot. *I. J. Humanoid Robot*. 9:1250017. doi: 10.1142/S021984361250017X

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., et al. (2014). "Caffe: convolutional architecture for fast feature embedding," in *ACM* (Orlando, FL).

Kingma, D., and Ba, J. (2015). "Adam: a method for stochastic optimization," in *ICLR* (San Diego, CA).

Kober, J., and Peters, J. R. (2009). "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems 21* (Vancouver, BC).

Lampe, T., and Riedmiller, M. (2013). "Acquiring visual servoing reaching and grasping skills using neural reinforcement learning," in *The 2013 International Joint Conference on Neural Networks (IJCNN)* (Dallas, TX).

Lange, S., and Riedmiller, M. (2010). Deep auto-encoder neural networks in reinforcement learning," in *IJCNN* (Barcelona), 1–8.

Law, J., Shaw, P., Lee, M., and Sheldon, M. (2014). From saccades to grasping: a model of coordinated reaching through simulated development on a humanoid robot. *IEEE Trans. Auton. Mental Dev*. 6, 93–109. doi: 10.1109/TAMD.2014.2301934

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res*. 17, 1334–1373. Available online at: http://dl.acm.org/citation.cfm?id=2946645.2946684

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. (2017). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Rob. Res*. 37, 421–436. doi: 10.1177/0278364917710318

Levine, S., Wagener, N., and Abbeel, P. (2015). "Learning contact-rich manipulation skills with guided policy search," in *ICRA* (Seattle, WA).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2016). "Continuous control with deep reinforcement learning," in *ICLR* (San Juan).

Metta, G., and Fitzpatrick, P. (2003). "Early integration of vision and manipulation," in *Adaptive Behavior special issue on Epigenetic Robotics* (Portland).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*. 518, 529–533. doi: 10.1038/nature14236

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2017). Overcoming exploration in reinforcement learning with demonstrations. *CoRR*. Available online at: https://arxiv.org/abs/1709.10089 (Accessed Jun 06, 2018).

Nori, F., Natale, L., Sandini, G., and Metta, G. (2007). "Autonomous learning of 3D reaching in a humanoid robot," in *IROS* (San Diego, CA).

Popov, I., Heess, N., Lillicrap, T. P., Hafner, R., Barth-Maron, G., Vecerik, M., et al. (2017). Data-efficient Deep Reinforcement Learning for Dexterous

Manipulation. *CoRR*. Available online at: https://arxiv.org/abs/1704.03073 (Accessed Jun 06, 2018).

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., et al. (2009). "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software* (Kobe).

Rajeswaran, A., Lowrey, K., Todorov, E. V., and Kakade, S. M. (2017). "Towards generalization and simplicity in continuous control," in *Advances in NIPS* (Long Beach, CA).

Riedmiller, M. A., Hafner, R., Lampe, T., Neunert, M., Degrave, J., de Wiele, T. V., et al. (2018). Learning by playing-solving sparse reward tasks from scratch. *CoRR*. Available online at: https://arxiv.org/abs/1802.10567 (Accessed Jun 06, 2018).

Sermanet, P., Xu, K., and Levine, S. (2017). "Unsupervised perceptual rewards for imitation learning," in *Robotics: Science and Systems* (Cambridge, MA).

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). "Deterministic policy gradient algorithms," in *ICML* (Beijing).

Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.* 6, 215–219. doi: 10.1162/neco.1994.6.2.215

Tsurumine, Y., Cui, Y., Uchibe, E., and Matsubara, T. (2017). "Deep dynamic policy programming for robot control with raw images," in *IROS* (Vancouver, BC).

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 02 October 2018
doi: 10.3389/fnbot.2018.00055

Exploring Criticality as a Generic Adaptive Mechanism

Miguel Aguilera[1,2]* *and Manuel G. Bedia*[2,3]

[1] *Information and Autonomous Systems-Research Center for Life, Mind, and Society, University of the Basque Country, Donostia, Spain,* [2] *Department of Computer Science and Systems Engineering, University of Zaragoza, Zaragoza, Spain,* [3] *Interactive Systems, Adaptivity, Autonomy and Cognition Lab, Aragón Institute of Engineering Research, University of Zaragoza, Zaragoza, Spain*

The activity of many biological and cognitive systems is not poised deep within a specific regime of activity. Instead, they operate near points of critical behavior located at the boundary between different phases. Certain authors link some of the properties of criticality with the ability of living systems to generate autonomous or intrinsically generated behavior. However, these claims remain highly speculative. In this paper, we intend to explore the connection between criticality and autonomous behavior through conceptual models that show how embodied agents may adapt themselves toward critical points. We propose to exploit maximum entropy models and their formal descriptions of indicators of criticality to present a learning model that drives generic agents toward critical points. Specifically, we derive such a learning model in an embodied Boltzmann machine by implementing a gradient ascent rule that maximizes the heat capacity of the controller in order to make the network maximally sensitive to external perturbations. We test and corroborate the model by implementing an embodied agent in the Mountain Car benchmark test, which is controlled by a Boltzmann machine that adjusts its weights according to the model. We find that the neural controller reaches an apparent point of criticality, which coincides with a transition point of the behavior of the agent between two regimes of behavior, maximizing the synergistic information between its sensors and the combination of hidden and motor neurons. Finally, we discuss the potential of our learning model to answer questions about the connection between criticality and the capabilities of living systems to autonomously generate intrinsic constraints on their behavior. We suggest that these "critical agents" are able to acquire flexible behavioral patterns that are useful for the development of successful strategies in different contexts.

Keywords: criticality, learning, boltzmann machine, Ising model, heat capacity

OPEN ACCESS

Edited by:
Gianluca Baldassarre,
Consiglio Nazionale Delle Ricerche (CNR), Italy

Reviewed by:
Daniel Polani,
University of Hertfordshire, United Kingdom
Georg Martius,
Max-Planck-Institut für Intelligente Systeme, Germany

****Correspondence:***
Miguel Aguilera
sci@maguilera.net

Received: *09 January 2018*
Accepted: *20 August 2018*
Published: *02 October 2018*

Citation:
Aguilera M and Bedia MG (2018) Exploring Criticality as a Generic Adaptive Mechanism. Front. Neurorobot. 12:55. doi: 10.3389/fnbot.2018.00055

1. INTRODUCTION

In the field of cognitive science, the interest in developing models of intrinsic motivation is unquestionable. The practical uses are related to the possibility of having more autonomous artifacts. In recent years, a significant number of models and cognitive architectures have been developed in the literature, pursuing various methods to get better intrinsically motivated machines. However, most of these studies follow *ad hoc* rules or present many conceptual weaknesses (Oudeyer and Kaplan, 2009). Therefore, it is a major research challenge to find new methods for designing intrinsically motivated systems.

One of the most intriguing intuitions in this field is the one that considers that the best way for machines to acquire skills completely on their own (and useful to pursue goals) is by exploiting the sensorimotor patterns that they create during their body-environment interactions. In this sense, they would be able to quickly construct more complex behaviors using a second level of learning from these patterns, so that they could combine typical random exploration with goal-free exploration, handling useful information obtained during their interactions with the world.

This insight, initially proposed in Juarrero (1999), is the complete opposite of the one that exists in artificial intelligence. The traditional way of dealing with notions such as motivations, autonomous goals, or intentional behavior implicitly assumes a reductionist perspective about how the mind causes behaviors. Engineers basically program "desires" within the artifacts as specific instructions that are extrinsic to them. Considering the fact that the actions generated from prior mental entities imply the existence of a "homuncular assumption" underlying every action. In this sense, intentional behavior and its causes could be better understood as a dynamical process that takes shape through the interactions between organisms and their environments (Buhrmann et al., 2013). Thus, intentionality could be described from a complex dynamical perspective that raises profound implications in relation to the notions of causation and intention in human action. This changes the view of how the intentions that "one feels" exist as independent mental events, proposing instead a new perspective where they result from the self-organizing tendencies in the human-environment system. Intentions to act, from this perspective, are best characterized as dynamical processes that are embedded in the physical and social history of a cognitive agent and are constrained to the set of limited alternatives within the self-organized space that is defined for particular agent-environment interactions. This idea has been followed by several authors and has been exploited in the field of autonomous robotics. In particular, some progresses have been made in measuring how a robot could manage the information it receives by applying information theory metrics (Der et al., 2008; Martius et al., 2013; Wissner-Gross and Freer, 2013).

In this paper, we are interested in developing models with intrinsic motivations that are generated through the exploitation of the information in sensorimotor patterns. In particular, we are interested in designing an embodied agent that generates complex behavior by adapting to operate near critical points. Criticality is a ubiquitous phenomenon in nature, both in physical and biological systems. It refers to a distinctive set of properties that are found at the boundary that separates regimes with different dynamics: the transition between an ordered phase and a disordered phase. Some of these properties include (i) power-law divergences of some quantities that are described by critical exponents and (ii) maximal sensitivity to external perturbations (Salinas, 2001a,b). With regard to our interests, it is known that self-organizing properties that allow us to characterize "modes of critical behavior" are related to different functional domains of cognitive activity (Van Orden et al., 2003, 2012; Dixon et al., 2012). This leads us to think that criticality may be functionally useful in problem solving.

Most of the systems near critical points exhibit a wide range of time scales in their dynamics, being maximally responsive to certain external signals. For a system facing a problem, critical states leave open different courses of action (configured within a global state that is acutely context sensitive) that can be simultaneously constrained in only one course of action in an effective way. Hoffmann and Payton (2018) showed that this type of self-organizing critical processes can even be used to solve optimization problems with many local minima, in a more efficient way when compared to other random search methods.

It has also been conjectured that systems that show intentional behavior should self-organize into critical states (Van Orden and Holden, 2002; Van Orden et al., 2003), but, nevertheless, the connection between self-organized criticality and intrinsically generated behavior remains highly speculative. In general, although evidence of criticality has been found using different experimental methods, the connection between these indicators and the properties of mechanistic models of critical activity is thin (Wagenmakers et al., 2012). This makes it difficult to assess the connection between criticality and other cognitive phenomena, other than at the level of pure analogy. Interestingly, in the past few years, large sets of biological data have allowed the characterization–using maximum entropy models–of how the behaviors of different biological systems (e.g., networks of neurons, antibody segments, or flocks of birds) are poised near a critical point within their parameter space (Mora and Bialek, 2011; Tkacik et al., 2015). This has been a great step toward the development of deeper theoretical principles that lie behind the behavior of biological and cognitive systems. However, going beyond the importance of these models in explaining the emergence of criticality in specific experimental data, we propose a complementary perspective to address the development of "conceptual models" to explain how organisms are driven toward critical behavior at a more abstract level and what the behavioral correlates are when the agent adapts to critical points.

With all of the above information, in this paper, we seek to develop a mechanism that combines these two concepts: criticality and models of intrinsic motivation. In the study of cognitive processes, criticality always appears to be entangled with other features of adaptive behavior (e.g., perception, prediction, learning) in agents that interact with complex environments. Here, we use conceptual modeling that allows us to study this relationship in a neutral and abstract way.

Therefore, the aim of this paper is to propose a model that is able to drive synthetic agents toward critical points to potentially clarify what the contribution of criticality is in different contexts. Instead of making assumptions about the underlying dynamics of the elements of the agent's controller or a fine-tuning of the parameters of the system, our approach makes use of concepts from statistical mechanics to exploit macroscopic variables that drive the system to transition points between qualitatively different regimes of behavior. Some authors have studied the computational capabilities of recurrent neural networks that operate near the edge of chaos, that is, the transition from ordered to chaotic dynamics (Bertschinger and Natschläger, 2004). Here, we propose a conceptual model (a Boltzmann machine) that allows us to exploit its statistical properties to derive a learning rule to drive an embodied agent toward critical points of its parameter space. Maximum entropy models have the advantage of providing a formal description of the statistical distribution of the system, which we will exploit to characterize the indices of criticality and to derive rules that maximize these indices. An abstract mechanism that drives the agents to near critical points in different scenarios may help in understanding what the contributions of criticality "by itself" are or how criticality is related to other phenomena.

The paper is organized as follows. First, we introduce a Boltzmann machine as the simplest statistical mechanics model of pairwise correlations between elements of a network and, then, derive a learning model for driving the system toward critical points. The model exploits the heat capacity of the system, a macroscopic measure that works as a proxy for criticality (when the heat capacity diverges, a Boltzmann machine is at a critical point). Consequently, we test our learning model in an embodied agent that controls a Mountain Car (a classic reinforcement learning test bed) by finding that it is able to drive both the neural controller and the behavior of the agent to a transition point in the parameter space between qualitatively different behavioral regimes. Finally, we discuss the possible applications of our model to contribute to the development of deeper principles that govern biological and cognitive systems.

2. DRIVING A NEURAL CONTROLLER TOWARD A CRITICAL POINT

We propose a learning model for adjusting the parameters of a Boltzmann machine in order to drive the system near states of criticality. We take advantage of the fact that, at critical points, derivatives of thermodynamic quantities such as entropy may diverge (Mora and Bialek, 2011). An example of this is heat capacity, whose divergence is a sufficient condition for criticality (though not a necessary one). As discussed below, the heat capacity of a system is related to the derivative of the entropy of the system. If the heat capacity of the system diverges at a critical point, this means that the system is maximally sensitive to external perturbations, since very small perturbations push the system into order or disorder. We hypothesize that actively seeking to poise a system near a critical point may constitute an intrinsic mechanism to adapt to different environments and generate complex behaviors in different contexts.

We define our model as a stochastic artificial neural network (i.e., a Boltzmann machine) (Ackley et al., 1985) that follows a maximum entropy distribution:

$$P(\sigma) = \frac{1}{Z} exp\left[\beta \sum_i h_i\sigma_i + \sum_{i<j} J_{ij}\sigma_i\sigma_j\right] \qquad (1)$$

where the distribution follows an exponential family $P(\sigma) = \frac{1}{Z}e^{-\beta E(\sigma)}$, Z is a normalization value, the energy $E(\sigma)$ of each state is defined in terms of the bias h_i and symmetrical couplings J_{ij} between pairs of units, and $\beta = 1/(Tk_B)$, k_B is Boltzmann's constant and T is the temperature of the system.

Throughout the paper, we simulate the network that updates its state by applying Glauber dynamics to all the units within the network in a sequential random order at each simulation step. Glauber dynamics define the probability of the next state of a neuron i as

$$P(\sigma_i'|\sigma) = \frac{1}{1+e^{-2\beta\sigma_i' H_i}}, \quad H_i = h_i + \sum_j J_{ij}\sigma_j \qquad (2)$$

where s is the state of the system at time t and s' is the state at time $t+1$.

In order to know if the system is near a critical point, typically, the divergence of certain quantities is measured. One of these quantities is the heat capacity of the system, which is generally defined as

$$C(\sigma) = -\beta\frac{\partial S(\sigma)}{\partial \beta} = \beta^2\left(\langle E(\sigma)^2\rangle - \langle E(\sigma)\rangle^2\right) \qquad (3)$$

where $S(\sigma) = -\sum_\sigma P(\sigma)\log(P(\sigma))$, and the heat capacity is defined in terms of the global energy of the system $E(\sigma) = -\sum_i h_i\sigma_i - \sum_{i<j} J_{ij}\sigma_i\sigma_j$, making it impractical to derive learning rules based on local information. This will be important for applying our learning rule to an embodied agent, where the energy of the states of the environment is not directly accessible to the system. Instead, we can find a more tractable indicator of criticality by defining the heat capacity of the system from the conditional entropy of each neuron that depicts transitions between states. We define conditional entropy of a neuron i as

$$\begin{aligned} S(\sigma_i'|\sigma) &= -\sum_\sigma P(\sigma)\sum_{\sigma_i'}\log(P(\sigma_i'|\sigma))\cdot P(\sigma_i'|\sigma) \\ &= -\sum_\sigma P(\sigma)\left(\beta H_i\tanh(\beta H_i) - \log(2\cosh(\beta H_i)\right) \qquad (4) \end{aligned}$$

Thus, we define the heat capacity associated with the conditional entropy of neuron i as

$$\begin{aligned} C(\sigma_i'|\sigma) &= -\beta\frac{\partial S(\sigma_i'|\sigma)}{\partial\beta} = \sum_\sigma P(\sigma)\left(\frac{H_i^2\beta^2}{\cosh(\beta H_i)^2}\right. \\ &\left. -\beta\left(E(\sigma) - \langle E(\sigma)\rangle\right)(\beta H_i\tanh(\beta H_i) - \log(2\cosh(\beta H_i))\right) \end{aligned} \qquad (5)$$

which still contains terms that are dependent on the global energy of the system $E(\sigma)$. In order to derive a learning rule based only on local information, we can introduce individual temperatures T_i for each neuron, which are associated with an individual inverse temperature β_i. In other words, instead of modifying the temperature of the system as a whole, we introduce the possibility of modifying "individual temperatures." We use these quantities to derive a simplified version of the heat capacity that can be

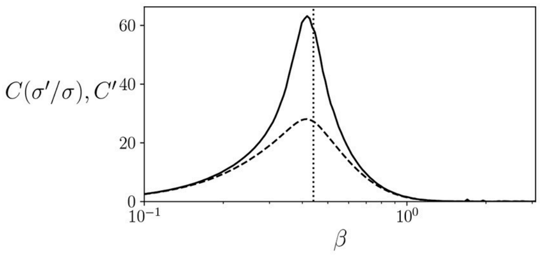


FIGURE 1 | Values of $C(\sigma_i'|\sigma)$ (solid line) and C' (dashed line) for an 8 × 8 lattice Ising model at different temperatures. We find that both quantities show a peak around the critical temperature at $\beta = \log(1+\sqrt{2})/2$ (dotted line).

computed as an average of a function that is defined only by local variables,

$$C_i' = -\beta_i \frac{\partial S(\sigma'|\sigma)}{\partial \beta_i} = \sum_{\sigma} P(\sigma) \left(\frac{H_i^2 \beta_i^2}{\cosh(\beta_i H_i)^2} + \beta_i \left(\sigma_i H_i - \langle \sigma_i H_i \rangle\right) \left(\beta_i H_i \tanh(\beta_i H_i) - \log(2\cosh(\beta_i H_i)\right) \right) \quad (6)$$

and we compute the total approximated heat capacity as $C' = \sum_i C_i'$. Note that the properties of C' and $C(\sigma_i'|\sigma)$ can be different, since we are neglecting the terms that reflect global interactions, retaining only local interactions. However, we argue that the simplified C' may be a suitable indicator for driving a system to a critical point.

As an example, we can compute the values of $C(\sigma_i'|\sigma)$ and C' for the well known Ising model in a rectangular lattice, for which the critical temperature is known to be at $T = 2k_B/\log(1+\sqrt{2})$ for an infinite-size system. Note that, since C and C' can be defined as an average of the terms by multiplying $P(\sigma)$ in equations, in practice it can be approximated by running a system for several steps and computing the mean value of these terms. We simulate an 8 × 8 periodic square lattice during 100,000 simulation steps. As seen in **Figure 1**, both $C(\sigma_i'|\sigma)$ and C' have a peak around the critical temperature, although the peak is more pronounced in the case of $C(\sigma_i'|\sigma)$, showing how, at least in some cases, a peak in C' can be an indicator of proximity to a critical point.

An earlier study (Aguilera and Bedia, 2018) used indirect indicators as the distribution of correlations of the system to find points of divergence of the heat capacity. Here, we try to directly maximize the heat capacity by obtaining the relation between parameter changes and the heat capacity. We define a learning rule that adjusts the values of h_i and J_{ij} by using a gradient ascent rule that maximizes the value of the simplified heat capacity C', with the intention of driving the system to critical points that are depicted by a singularity of the heat capacity.

In order to simplify the notation, we define the quantities $F_i = H_i \tanh(H_i) - \log(2\cosh(H_i)$, $G_i = \frac{H_i^2}{\cosh(H_i)^2} + \sigma_i H_i F_i$, and $K_i = \langle \sigma_i H_i \rangle$. Using the derivatives of the probability distribution in Equation 1, $\frac{\partial P(\sigma)}{\partial h_i} = (\sigma_i - \langle \sigma_i \rangle) P(\sigma)$ and $\frac{\partial P(\sigma)}{\partial J_{ij}} = (\sigma_i\sigma_j - \langle \sigma_i \sigma_j \rangle) P(\sigma)$, and the derivatives of F_i, G_i, and K_i, we derive the learning rule that ascends the gradient of C_i' and drives the system toward critical points as

$$\begin{aligned}
\frac{\partial C_i'}{\partial h_i} &= \langle \frac{\partial G_i}{\partial h_i} \rangle + \langle \sigma_i G_i \rangle - \langle \sigma_i \rangle \langle G_i \rangle - \frac{\partial K_i}{\partial h_i} \langle F_i \rangle - K_i (\langle \frac{\partial F_i}{\partial h_i} \rangle \\
&\quad + \langle \sigma_i F_i \rangle - \langle \sigma_i \rangle \langle F_i \rangle) \\
\frac{\partial C_i'}{\partial J_{ij}} &= \langle \frac{\partial G_i}{\partial J_{ij}} \rangle + \langle \sigma_i \sigma_j G_i \rangle - \langle \sigma_i \sigma_j \rangle \langle G_i \rangle - \frac{\partial K_i}{\partial J_{ij}} \langle F_i \rangle - K_i (\langle \frac{\partial F_i}{\partial J_{ij}} \rangle \\
&\quad + \langle \sigma_i \sigma_j F_i \rangle - \langle \sigma_i \sigma_j \rangle \langle F_i \rangle)
\end{aligned} \quad (7)$$

where

$$\begin{aligned}
\frac{\partial F_i}{\partial h_i} &= \frac{H_i}{\cosh(H_i)^2}, \\
\frac{\partial F_i}{\partial J_{ij}} &= \frac{H_i \sigma_j}{\cosh(H_i)^2}, \\
\frac{\partial G_i}{\partial h_i} &= \frac{2H_i(1 - H_i \tanh(H_i))}{\cosh(H_i)^2} + \sigma_i F_i + \sigma_i H_i \frac{\partial F_i}{\partial h_i}, \\
\frac{\partial G_i}{\partial J_{ij}} &= \frac{2H_i \sigma_j (1 - H_i \tanh(H_i))}{\cosh(H_i)^2} + \sigma_i \sigma_j F_i + \sigma_i H_i \frac{\partial F_i}{\partial J_{ij}}, \\
\frac{\partial K_i}{\partial h_i} &= \langle \sigma_i \rangle + \langle \sigma_i^2 H_i \rangle - \langle \sigma_i \rangle K_i \\
\frac{\partial K_i}{\partial J_{ij}} &= \langle \sigma_i \sigma_j \rangle + \langle \sigma_i^2 \sigma_j H_i \rangle - \langle \sigma_i \sigma_j \rangle K_i
\end{aligned} \quad (8)$$

In the following section, we use this learning rule to drive the neural controller of an embodied agent toward a critical point. In order to do so, we need to take into account the environment at the time of learning. If we consider two interconnected Boltzmann machines (one being the neural controller and the other being the environment), Equation 7 holds perfectly, and we could design an adaptive controller that applies the rule to the values of i and j that correspond to units of the neural controller. In our case, the environment is not composed of units of a Boltzmann machine. Instead, we connect the Boltzmann machine of the neural controller to an environment that is defined as a classic example from reinforcement learning. Therefore, our learning rule will be valid as long as the statistics of the environment can be approximated by a Boltzmann machine with a sufficiently large number of units. Luckily, Boltzmann machines are universal approximators (Montúfar, 2014).

3. EMBODIED MODEL: MOUNTAIN CAR

In order to evaluate the behavior of the proposed learning model, we tested it in the Mountain Car environment (Moore, 1990). This environment is a classical test bed in reinforcement learning that depicts an underpowered car that must drive up a steep hill (**Figure 2**). Since gravity is stronger than the car's engine, the vehicle must learn to leverage its potential energy by driving to the opposite hill before the car is able to make it to the goal at the top. We simulated the environment by using the OpenAI Gym toolkit (Brockman et al., 2016). In this environment, the horizontal position x of the car is limited to an interval of

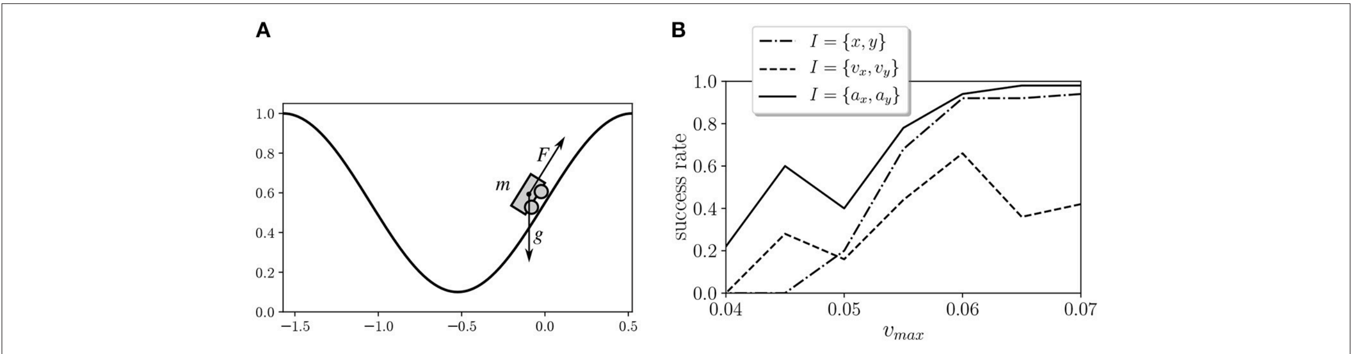


FIGURE 2 | (A) Mountain Car environment test from the OpenAI Gym toolkit. An underpowered car that must drive up a steep hill by balancing itself to gain momentum. **(B)** Success rates of the Mountain Car environment with different kinds of sensors.

$[-1.5\pi/3, 0.5\pi/3]$, and the vertical position of the car is defined as $y = 0.5(1 + \sin(3x))$. The velocity in the horizontal axis is updated at each time step as $v_x(t+1) = v_x(t) + 0.001a - 0.0025\cos(3x)$, where a is the action of the motor, which can be either -1, 0, or 1. The maximum velocity of the car is limited to an absolute value of v_{max}.

We defined the neural controller of the car as a fully-connected Boltzmann machine (without hidden neurons) that contains six sensors and six neurons. Initially, we tested different options of input: position, speed, and acceleration. For each input, the value is separated into its horizontal and vertical components, each input is discretized as arrays of three bits. Each sensor unit is assigned a value of 1 if its corresponding bit is active and a value of -1 otherwise. Two of the car neurons are connected to the motors, defined as $a = 1$ if both neurons are active, $a = -1$ if both neurons are inactive, and $a = 0$ otherwise.

In order to find critical points with maximum heat capacity, we propose a learning rule intended to climb the gradient defined by Equation 7 at a rate μ. Also, in order to avoid overfitting, we add an L2 regularization term λ penalizing large values of h_i and J_{ij} the parameters of the system. Finally, the learning rule is described as:

$$\begin{aligned} h_i &\leftarrow h_i + \mu \frac{\partial C_i'}{\partial h_i} - \lambda h_i \\ J_{ij} &\leftarrow J_{ij} + \mu \frac{\partial C_i'}{\partial J_{ij}} - \lambda J_{ij} \end{aligned} \qquad (9)$$

where $\mu = 0.02$, $\lambda = 0.002$, and $\frac{\partial C_i'}{\partial h_i}$ and $\frac{\partial C_i'}{\partial J_{ij}}$ are the result of Equation 7. The rule is applied to 20 different agents. Agents are initialized in the starting random position of the environment. Hidden and motor neurons are randomized, and the initial parameters h and J are sampled from a uniform random interval $[-0.01, 0.01]$. The agents are simulated for $1,000$ trials of $5,000$ steps, applying Equation 7 at the end of the trial for computing the values of $\frac{\partial C_i'}{\partial h_i}$ and $\frac{\partial C_i'}{\partial J_{ij}}$. Note that the agents are not reset at the end of the trial. After training, the values of h_i and J_{ij} are kept fixed for the rest of the analysis described in the paper.

We tested different types of inputs and values of v_{max}. The inputs tested were 1) the horizontal position and vertical position of the car $I = \{x, y\}$, 2) the horizontal speed and vertical speed of the car $I = \{v_x, v_y\}$, and 3) the horizontal acceleration and vertical acceleration of the car $I = \{a_x, a_y\}$. In all cases, horizontal and vertical values are discretized as arrays of three bits and are fed to the six sensor units. We tested seven values of v_{max} in the range $[0.04, 0.07]$ for the three types of inputs and the 20 agents, and we measured the success of the agents as their ability to reach the top of the agents in a trial of 50,000 steps after training (**Figure 2B**). In order to select a case where the task is feasible but not too easy, we chose $I = \{a_x, a_y\}$ and $v_{max} = 0.045$ for the experiments described below. The experimental results correspond to the 20 agents trained for this configuration.

4. RESULTS

In this section, we analyze the neural controller and the behavioral patterns of the agents in relation to the possibilities of their parameter space. In order to compare the agents with other behavioral possibilities, we explore the parameter space by changing the parameter β. Since the temperature of the model has no physical significance, modifying the value of β is equivalent to a global rescaling of the parameters of the agent that transforms $h_i \leftarrow \beta \cdot h_j$ and $J_{ij} \leftarrow \beta \cdot J_{ij}$, thus, exploring the parameter space along one specific direction. For 21 values of β that are logarithmically distributed in the interval $[10^{-1}, 10^{1}]$, we compute 20 agents for a trial of 10^6 simulation steps, after starting the agents from a random initial position (i.e., x in an interval $[0.4, 0.6]$) and a run of 10^4 simulation steps to avoid the initial transient. We use the results of these simulations for all the calculations in this section.

4.1. Signatures of Criticality in the Neural Controller

Firstly, we test whether the trained agents show signatures of critical behavior, looking for a Zipf's law in the probability distribution of the states of the neural controller and a peak in its heat capacity. In order to test that the criticality arises from the agent's configuration and not just from dynamics of the task, we compared the results of the trained agents with the 20 agents trained for maximizing the success in the task. In order to do so, we trained agents with a similar network by using a microbial genetic algorithm (Harvey, 2009) that maximizes the number of times an agent is able to climb the mountain during the 5,000 steps (reseting the agent after each climb). By counting the occurrence of each possible state of the 12 neurons of the agents (including sensor, hidden, and motor neurons), we can compute the probability distribution of the Boltzmann machine $P(\sigma)$.

We observed that all agents approximately follow a Zipf's law at $\beta = 1$ (**Figure 3A**) for almost three decades, which is a good agreement for the limited size of the system (note that the possible states of the system are limited to 2^{12} states). All trained agents showed a similar distribution close to Zipf's law. In comparison, agents maximized to solve the task failed to show a distribution that is consistent with Zipf's law.

Secondly, given that another indicator of critical points is the divergence of the heat capacity of the system, we estimated the heat capacity of hidden and motor neurons. From the data generated from the simulation, we can estimate $C(\sigma_i'|\sigma)$ by computing entropy $S(\sigma_i'|\sigma)$ from Equation 4. We use cubic interpolation for estimating the function of $S(\sigma_i'|\sigma)$ with respect to β and for estimating its derivative to compute $C(\sigma_i'|\sigma)$. We observe in **Figure 3B** that the heat capacity peaks at around the operating temperature (i.e., the temperature used during training, $\beta = 1$). This, together with the Zipf's distribution, suggests that the system is operating near a critical point. In comparison, agents maximized to solve the task fail to present a clear peak of the heat capacity at a specific temperature, indicating that no significant transitions are taking place.

4.2. Behavioral Transitions in the Parameter Space

What is implied when the agent drives its neural controller near a critical point? It should be remarked here that our agents are given no explicit goal. Instead, they only tend toward behavioral patterns that maximize the heat capacity of their neurons, independently of whether this behavior enables them to reach the top of the mountain or not (in fact, only 12 of the 20 trained agents are able to climb to the top of the mountain). In relation to this, we explore the effects of transiting the critical point by observing the different behavioral modes of the agent in the parameter space. The behavior of the car can be described just by the position x and the speed v at different moments of time.

In **Figures 4A–C**, we can observe the behavior of the car for $\beta = \{0.25, 1, 4\}$ for a specific agent, for an interval of $4,000$ simulation steps, after an initial run of $10,000$ steps to remove the initial transient. In this particular example, there is an asymmetry in the behavior of the car, which only climbs the left mountain. This asymmetry is provoked by the sign of the offset value h_i of motor units. If we compute the median and quartile values of y at the trial for each value of β (**Figure 4D**) we observe that, slightly below the operating temperature, there is a transition from not being able to reach the top of the mountains to those that are able to do so. Moreover, in all agents that are able to reach the top of the mountain, the results are similar. Out of the agents that are not able to reach the top, five display similar transitions in the median value of height y and the median absolute velocity v of the car. The remaining three agents fail to show a transition in median values of basic behavioral variables, although this does not preclude the possibility of another type of less evident behavioral transition.

What has changed in this behavioral transition? We are interested in knowing how these behavioral regimes are qualitatively different. We explore this issue by using information theory to characterize how different variables of the agent interact at different points of the parameter space. Specifically, we are interested in the relation between sensor, hidden, and motor neurons, which determines the behavior of the agent in its environment.

Are agents merely reactive to sensory inputs or is there a more complex interplay between sensor, hidden, and motor units? In order to answer this, we characterize the interaction between variables by using measures from information theory. First, we measured the values of entropy and mutual information between $S(X)$ and $I(X;Y)$, where

$$S(X) = -\sum_{x \in X} P(x) \log(P(x)) \tag{10}$$

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) \log(\frac{P(x,y)}{P(x)P(y)}) \tag{11}$$

and X and Y are random discrete variables. Entropy $S(X)$ measures the amount of information of each variable, whereas mutual information $I(X,Y)$ measures the amount of information that overlaps between two variables due to their mutual dependence. In **Figures 5A,B**, we observe the entropy and mutual information for the variables S, H, and M for different values of β.

By defining S, H, and M as the joint distributions of sensor, hidden, and motor neurons, respectively, we analyze how information is distributed among the three groups of variables. We observe how the entropy of the hidden neuron H decreases significantly in the transition around $\beta = 1$. Similarly, all values of mutual information, especially between motors M and sensors S, increase around $\beta = 1$. This suggests a transition from independent variables showing unconstrained information to highly correlated variables with high mutual dependencies.

We suspect that it is just in this transition point where an agent can maximize its interactive capacities, combining integration and segregation between variables. To check this, we use information decomposition (Timme et al., 2014) to compute synergies between variables X_1 and X_2 with respect to a third one Y, defined as

$$\Psi(Y;X_1,X_2) = I(Y;X_1,X_2) - I(Y;X_1) - I(Y;X_2) + I_{min}(Y;X_1,X_2) \tag{12}$$

where $I_{min}(Y;X_1,X_2)$, defined in Williams and Beer (2011), is the redundant information that X_1 and X_2 share about Y. The resulting synergy $\Psi(Y;X_1,X_2)$ is able to capture information about Y that is not available from either X_1 and X_2 alone but from their interaction (the classical example is the relation between the output and inputs of an XOR gate). In other words, the intention is to measure how much information emerges from the

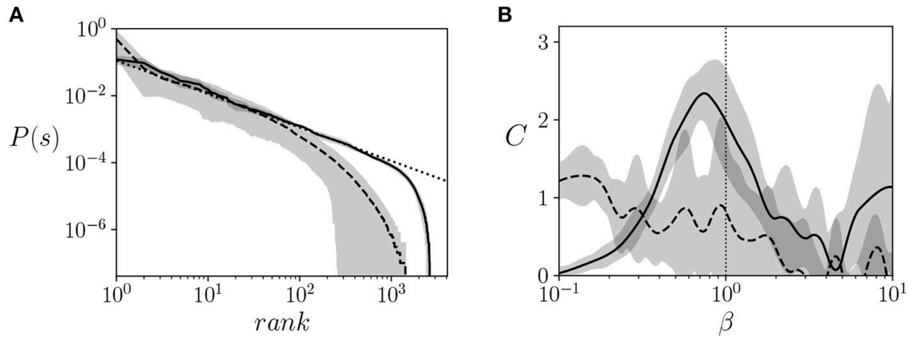


FIGURE 3 | Signatures of criticality. **(A)** Ranked probability distribution function of the neural controllers of trained agents (solid line) vs. a distribution that follows a Zipf's law, (i.e., $P(\sigma) = 1/rank$, dotted line) and the distribution of an agent optimized to solve the task (dashed line). We observe a good agreement between the model and the Zipf's law, suggesting critical scaling. **(B)** Heat capacity $C(\sigma_i'|\sigma)$ vs. β of trained agents (solid line), computed using Equation 4 for calculating the entropy and deriving a cubic interpolation of the entropy function with respect to β. The heat capacity of trained agents is compared with the heat capacity of agents tuned to maximize the ability to climb the mountain (dashed lines) The gray areas represent the error bars of the 20 agents for each value of β. The vertical dotted line specifies the value of $\beta = 1$ where agents operate during training.

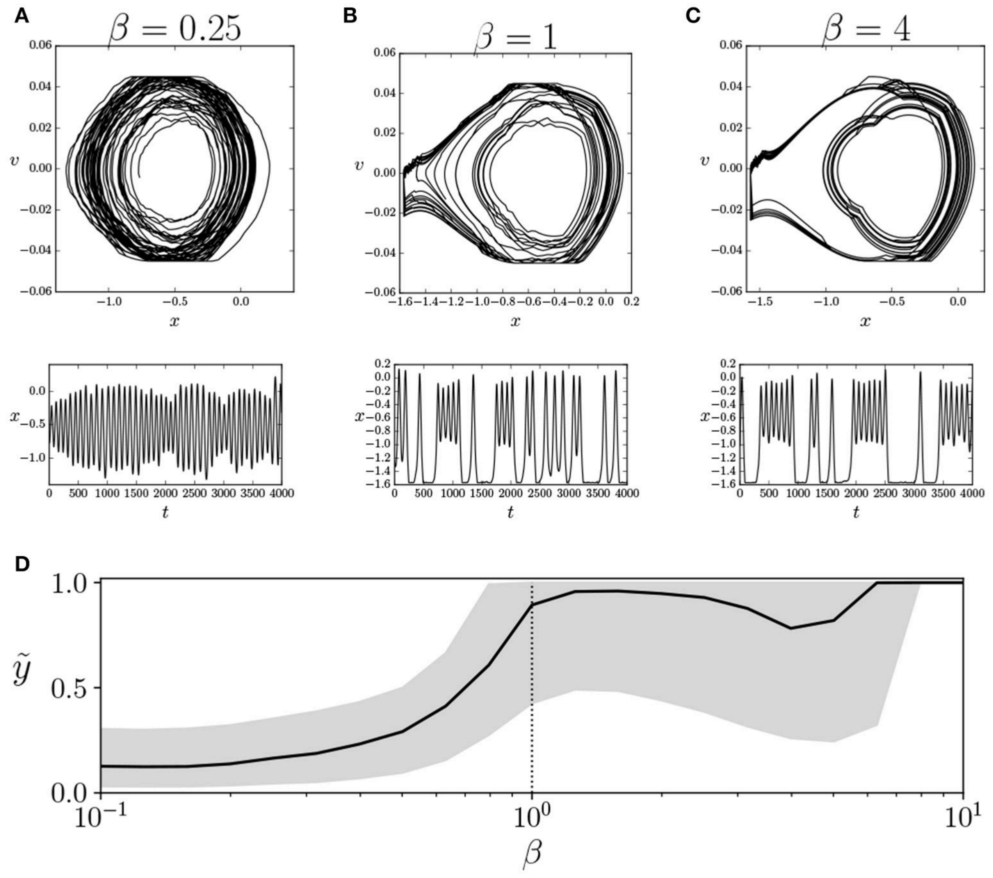


FIGURE 4 | Transition in behavioral regime of the agent. We show the behavior of an agent for an interval of 4,000 steps with values of β of 0.25 **(A)**, 1 **(B)**, and 4 **(C)**, depicting the trajectories of the car in its phase space (*x vs. v*, top) and the evolution of the values of *x* (bottom). We observe that $\beta = 1$ is a transition point between two modes of behavior. **(D)** Median vertical position of the car $\tilde{y}$ (solid line) and its upper and lower quartiles (gray area). We observe a transition near $\beta = 1$ where the agent reaches the top of the mountain. Similar transitions are identified in 12 of the 20 simulated agents. The vertical dotted line specifies the value of $\beta = 1$ where the agents operate during training.

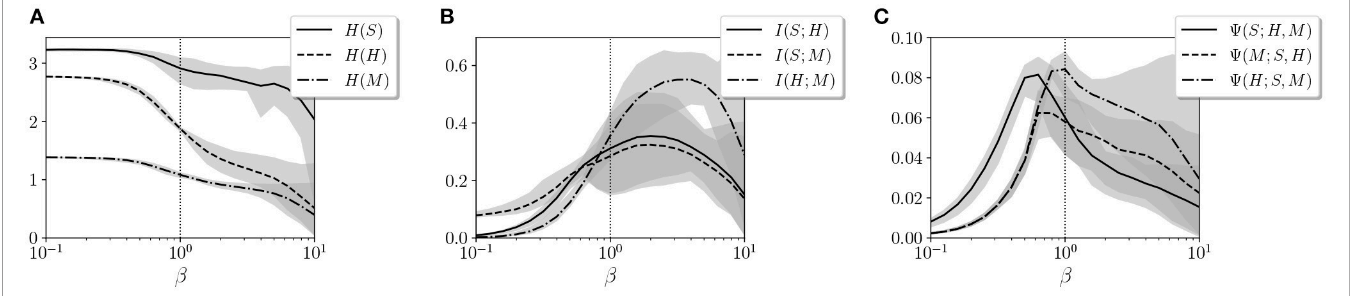


FIGURE 5 | Values of **(A)** entropy, **(B)** mutual information, and **(C)** synergistic information for variables S, H, and M. The gray areas represent the error bars of the 20 agents for each value of β. The vertical dotted line specifies the value of $\beta = 1$ where the agents operate during training.

interaction between variables instead of being contained in the variables alone.

As we observe in **Figure 5C**, the synergy $\Psi(S; H, M)$ between motor and hidden neurons about sensor information and the synergy $\Psi(M; S, H)$ between sensor and hidden neurons about sensor information peak at values of β lower than 1, while the synergies of motor neurons with sensor neurons, $\Psi(H; S, M)$ peak at larger values of β. Since the environment of the agent is completely deterministic, it seems adequate that larger values of β (i.e., less random behavior) more effectively transmit information from sensors to motors, while maximum interaction between hidden and other neurons occurs at a point with a lower β.

In conclusion, we propose a learning model that is designed to drive an embodied agent close to critical points in its parameter space, poising both the neural controller and the behavioral patterns of the agent near a transition point between qualitatively different regimes of operation. In the case of the neural controller, we find that the Boltzmann machine shows its peak capacity at a point around $\beta = 1$. Moreover, by measuring entropy and mutual information between groups of neural units, we find that the agent is poised at a transition point between a regime with high entropy but low coordination between units and a regime of high mutual entropy but low entropy. By analyzing the synergistic interaction between sensor, hidden, and motor units of the system, we find that interactions between groups of neural units are maximized for the operating temperature (although synergistic measures need to be taken into account carefully, since there is an ongoing debate around their formulation, Olbrich et al., 2015). We find a transition at a point slightly under $\beta = 1$, which also coincides with a point of transition between behavioral regimes in 12 of the 20 agents. These results suggest that the system may be exploiting a critical point for maximizing the interaction between the components of its neural controller, its sensory input, and its motor behavior.

5. DISCUSSION

The rule described here has some similarities and differences with respect to other work on maximizing information quantities in neural networks. Several measures have recently been introduced and have been demonstrated to be viable and powerful tools to express principles for driving autonomous systems. They are measures that are independent of the specific realization and domain invariants. We highlight, for example, predictive information measures (also called excess entropy or effective measure complexity) or methods that maximize entropy reinforcement learning (optimizing policies that maximize both the expected return and the expected entropy of the policy). With respect to the above mentioned measures, our idea differs in some aspects. On the one hand, predictive information is applied at the behavioral level of the whole system (Martius et al., 2013), whereas the learning rule that we propose is defined at the neuronal level (similar to local learning principles like the Hebb rule). Although our rule acts on the internal level, it is linked to information theoretic quantities on the behavioral level. On the other hand, the basic goal in conventional reinforcement learning algorithms is to maximize the expected sum of rewards combined with a more general maximum entropy objective (Ziebart et al., 2008). By contrast, in our method there is no reward that is maximized. Though such algorithms have been successfully used in a number of approaches, our method does not seek to optimize future rewards or maximize behaviors.

At this point, we can reflect on our original questions. Why do biological systems behave near criticality? What are the benefits for a biological system to move toward this special type of point? Also, more importantly, how can our learning model help answer these questions? By reviewing the relevant literature, one finds that interpretations of criticality are too speculative in general. For example, Beggs (2008) hypothesizes that neural systems that operate at a critical point can optimize information processing and its computational power. Mora and Bialek (2011) discuss the experimental evidence of criticality in a wide variety of systems and propose that criticality could provide better defense mechanisms against predators (in animals), gain selectivity in response to stimuli (in auditory systems, or improved mechanisms to anticipate attacks (in immunological systems). Nevertheless, the reasoning that gives support to these hypotheses is based more on generic suggestions than on scientifically rigorous statements. More detailed analyses are needed to test speculations, and our opinion is that conceptual models of embodied criticality in natural systems can usefully demonstrate how transition points in the parameter space of behavioral regimes can be found and exploited to obtain functional advantages such as those mentioned above. For this purpose, rather than citing specific biological instances of critical phenomena, we used an abstract framework for driving embodied agents to critical points. Our model approaches a critical point as well but remains at the disordered phase. Similar phenomena have been observed in biological neural networks (Priesemann et al., 2014). It has been proposed that information flow, generally, peaks on the disordered side of critical phase transitions (Barnett et al., 2013). More detailed studies of general mechanisms that drive a system to criticality may help shed light on these issues.

From the results obtained, what would be the main advantages of using an abstract model to study criticality? On the one hand, criticality generally appears to be entangled with other capabilities that are developed by biological systems, and interpretations about the advantages of criticality typically refer to tangible benefits for the system (e.g., at an evolutionary level, as the source of a new range of capabilities or better mechanisms for surviving in open environments, etc.); it is difficult to distinguish whether criticality is the cause or the consequence of such effects. On the other hand, we believe that the use of conceptual models such as those presented here allows a more intriguing hypothesis to be tested. For example, our general mechanism that drives an embodied neural controller to criticality has the potential to capture the contribution of criticality "by itself" to the behavior of adaptive agents in different scenarios, as well as the relationship between criticality and other biological and cognitive phenomena. Furthermore, the present model could be implemented in more complex embodied setups, for example, involving specific tasks of adaptive behavior that add environmental constraints (e.g., exploration, decision-making, categorical perception) or biological requirements (e.g., an internal metabolism or other biological drives such as hunger or thirst), and it could be used to observe how compliance with these biological and cognitive requirements interplays with the drive toward critical points in the neural controller of the agent. We could, thus, explore how criticality can contribute to the capabilities observed in natural organisms.

Finally, one of the most important conclusions we highlight is that systems at critical points can solve problems for which they were not programmed. This approach can be further linked to the analysis of particular features in animal behavior that are commonly interpreted without assuming a necessary pragmatic perspective of analysis. For example, the role of "play" in humans and other species. We observe certain similarities in the behavior of the developed embodied agent and the notion of play. In general, it is assumed that "solving a problem" is "being able to find a solution." In computational views of cognition, this requires handling representations of the world between which there is a configuration (the one in which the objective is reached) that the system must find. On the contrary, "play" is precisely not a problem requiring a solution. "Play" does not intend to solve a specific problem. Over time, "play" self-structures processes that are governed by the dialectics of expansion and contraction of possibilities. Its freedom lies in the capability of players to acquire and create novel nonarbitrary constraints during the processes involved (Di Paolo et al., 2010). We think that this may be a good metaphor for how the Mountain Car agent reaches the problem solution.

There are also other studies in the field of "play" that relate creativity, intrinsic motivations, and maximum entropy measurements. For example, Schmidhuber (2010) addresses the problem of how to model aspects of human player behavior that are not explained by either rational or goal-driven decision making behavior and without extrinsic reinforcement such as game score. The author focuses on analyzing the relationship between intrinsic motivation and creativity based on maximizing intrinsic reward for the active creation and the discovery of surprising patterns. In other exploratory studies (Guckelsberger et al., 2017), it is found that metrics such as empowerment can be useful to create specific models of intrinsic motivation in game design. Empowerment (Klyubin et al., 2005, 2008) quantifies the capacity of the agent to influence the world in a way that this influence is perceivable via the agent's sensors. These types of analyses (Roohi et al., 2018) identify the correlations between empowerment and challenge, attention, or engagement, by hypothesizing that maximum entropy measurements can also be used to create support tools for game designers.

In conclusion, we present, here, a model that does not address any particular task but solves a problem. It is interesting to note that it seems to exhibit intrinsic motivations but without being externally imposed, since its behavior is reduced to exploiting the criticality regime in which the system operates. Until now, the traditional study of criticality in living systems has rested on largely speculative grounds. The study of formal models and the increasing amount of high quality data together with advances in statistical mechanics models will make it possible to link experimental evidence and data-driven models with general conceptual models, paving the way for a rigorous exploration of the governing that lie behind the behavior of biological organisms in complex environments.

AUTHOR CONTRIBUTIONS

MA conceived and conducted the experiments. MA and MB analyzed the results and wrote the manuscript.

FUNDING

Research was supported in part by the Spanish National Programme for Fostering Excellence in Scientific and Technical Research project PSI2014-62092-EXP and by the project TIN2016-80347-R funded by the Spanish Ministry of Economy and Competitiveness. MA was supported by the UPV/EHU postdoctoral training program ESPDOC17/17.

REFERENCES

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cogn. Sci.* 9, 147–169.

Aguilera, M., and Bedia, M. G. (2018). Adaptation to criticality through organizational invariance in embodied agents. *Sci. Rep.* 8:7723. doi: 10.1038/s41598-018-25925-4

Barnett, L., Lizier, J. T., Harré, M., Seth, A. K., and Bossomaier, T. (2013). Information flow in a kinetic ising model peaks in the disordered phase. *Phys. Rev. Lett.* 111:177203. doi: 10.1103/PhysRevLett.111.177203

Beggs, J. M. (2008). The criticality hypothesis: how local cortical networks might optimize information processing. *Philos. Trans. R. Soc. Lond. A Mathe. Phys. Eng. Sci.* 366, 329–343. doi: 10.1098/rsta.2007.2092

Bertschinger, N., and Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural Comput.* 16, 1413–1436. doi: 10.1162/089976604323057443

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). OpenAI gym. *arXiv:1606.01540* [Preprint].

Buhrmann, T., Paolo, D., Alejandro, E., and Barandiaran, X. (2013). A dynamical systems account of sensorimotor contingencies. *Front. Psychol.* 4:285. doi: 10.3389/fpsyg.2013.00285

Der, R., Güttler, F., and Ay, N. (2008). "Predictive information and emergent cooperativity in a chain of mobile robots," in *The Eleventh International Conference on the Simulation and Synthesis of Living Systems* (Cambridge, MA; Winchester, UK: MIT Press), 166–172.

Di Paolo, E. A., Rohde, M., and De Jaegher, H. (2010). "Horizons for the enactive mind: values, social interaction, and play," in *Enaction: Toward a New Paradigm for Cognitive Science*, eds J. Stewart, O. Gapenne, and E. Di Paolo (Cambridge, MA: MIT Press), 33–87.

Dixon, J. A., Holden, J. G., Mirman, D., and Stephen, D. G. (2012). Multifractal dynamics in the emergence of cognitive structure. *Top. Cogn. Sci.* 4, 51–62. doi: 10.1111/j.1756-8765.2011.01162.x

Guckelsberger, C., Salge, C., Gow, J., and Cairns, P. (2017). "Predicting player experience without the player.: an exploratory study," in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '17, (New York, NY: ACM), 305–315.

Harvey, I. (2009). "The microbial genetic algorithm," in *Proceedings of the European Conference on Artificial Life* (Budapest: Springer), 126–133.

Hoffmann, H., and Payton, D. W. (2018). Optimization by self-organized criticality. *Sci. Rep.* 8:2358 doi: 10.1038/s41598-018-20275-7

Juarrero, A. (1999). *Dynamics in Action: Intentional Behavior as a Complex System: Alicia Juarrero: 9780262600477: Amazon.com: Books*. Cambridge, MA: MIT Press.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). Empowerment: a universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, Vol.1. 128-135.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2008). Keep your options open: an information-based driving principle for sensorimotor systems. *PLoS ONE* 3:e4018. doi: 10.1371/journal.pone.0004018

Martius, G., Der, R., and Ay, N. (2013). Information driven self-organization of complex robotic behaviors. *PLoS ONE* 8:e63400. doi: 10.1371/journal.pone.0063400

Montúfar, G. F. (2014). Universal approximation depth and errors of narrow belief networks with discrete units. *Neural Comput.* 26, 1386–1407. doi: 10.1162/NECO_a_00601

Moore, A. W. (1990). *Efficient Memory-Based Learning for Robot Control.* Technical Report, University of Cambridge, Computer Laboratory.

Mora, T., and Bialek, W. (2011). Are biological systems poised at criticality? *J. Stat. Phys.* 144, 268–302. doi: 10.1007/s10955-011-0229-4

Olbrich, E., Bertschinger, N., and Rauh, J. (2015). Information decomposition and synergy. *Entropy* 17, 3501–3517. doi: 10.3390/e17053501

Oudeyer, P.-Y., and Kaplan, F. (2009). What is intrinsic motivation? A typology of computational approaches. *Front. Neurorobotics*, 1:6. doi: 10.3389/neuro.12.006.2007

Priesemann, V., Wibral, M., Valderrama, M., Pröpper, R., Le Van Quyen, M., Geisel, T., et al. (2014). Spike avalanches *in vivo* suggest a driven, slightly subcritical brain state. *Front. Syst. Neurosci.* 8:108. doi: 10.3389/fnsys.2014.00108

Roohi, S., Takatalo, J., Guckelsberger, C., and Hämäläinen, P. (2018). "Review of intrinsic motivation in simulation-based game testing," in *CHI '18 Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, Vol. 347 (New York, NY: ACM), 1–347.

Salinas, S. R. A. (ed.). (2001a). "Phase transitions and critical phenomena: classical theories," in *Introduction to Statistical Physics, Graduate Texts in Contemporary Physics* (New York, NY: Springer), 235–256.

Salinas, S. R. A. (2001b). "Scaling theories and the renormalization group," in *Introduction to Statistical Physics* (New York, NY: Springer New York), 277–304.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990 #x2013;2010). *IEEE Trans. Auton. Mental Dev.* 2, 230–247. doi: 10.1109/TAMD.2010.2056368

Timme, N., Alford, W., Flecker, B., and Beggs, J. M. (2014). Synergy, redundancy, and multivariate information measures: an experimentalist's perspective. *J. Comput. Neurosci.* 36, 119–140. doi: 10.1007/s10827-013-0458-4

Tkačik, G., Mora, T., Marre, O., Amodei, D., Palmer, S. E., Berry, M. J., et al. (2015). Thermodynamics and signatures of criticality in a network of neurons. *Proc. Natl. Acad. Sci. U.S.A.* 112, 11508–11513. doi: 10.1073/pnas.1514188112

Van Orden, G., Hollis, G., and Wallot, S. (2012). The blue-collar brain. *Fractal Physiol.* 3:207. doi: 10.3389/fphys.2012.00207

Van Orden, G. C., and Holden, J. G. (2002). Intentional contents and self-control. *Ecol. Psychol.* 14, 87–109. doi: 10.1080/10407413.2003.9652753

Van Orden, G. C., Holden, J. G., and Turvey, M. T. (2003). Self-organization of cognitive performance. *J. Exp. Psychol. Gen.* 132, 331–350. doi: 10.1037/0096-3445.132.3.331

Wagenmakers, E. J., van der Maas, H. L. J., and Farrell, S. (2012). Abstract concepts require concrete models. *Top. Cogn. Sci.* 4, 87–93. doi: 10.1111/j.1756-8765.2011.01164.x

Williams, P. L., and Beer, R. D. (2011). Generalized measures of information transfer. *arXiv:1102.1507 [Preprint]*.

Wissner-Gross, A. D., and Freer, C. E. (2013). Causal entropic forces. *Phys. Rev. Lett.* 110, 168702. doi: 10.1103/PhysRevLett.110.168702

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. (2008). "Maximum entropy inverse reinforcement learning," in *Proceedings of the National Conference on Artificial Intelligence,* Vol. 3 (Amsterdam: IOS Press), 1433–1438.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH
published: 09 October 2018
doi: 10.3389/fnbot.2018.00063

Intrinsic Rewards for Maintenance, Approach, Avoidance, and Achievement Goal Types

Paresh Dhakan[1]*, *Kathryn Merrick*[2], *Iñaki Rañó*[3] *and Nazmul Siddique*[1]

[1] Intelligent Systems Research Centre, Ulster University, Derry, United Kingdom, [2] School of Engineering and Information Technology, University of New South Wales, Canberra, ACT, Australia, [3] Embodied Systems for Robotics and Learning, The Mærsk Mc Kinney Møller Institute, University of Southern Denmark, Odense, Denmark

In reinforcement learning, reward is used to guide the learning process. The reward is often designed to be task-dependent, and it may require significant domain knowledge to design a good reward function. This paper proposes general reward functions for maintenance, approach, avoidance, and achievement goal types. These reward functions exploit the inherent property of each type of goal and are thus task-independent. We also propose metrics to measure an agent's performance for learning each type of goal. We evaluate the intrinsic reward functions in a framework that can autonomously generate goals and learn solutions to those goals using a standard reinforcement learning algorithm. We show empirically how the proposed reward functions lead to learning in a mobile robot application. Finally, using the proposed reward functions as building blocks, we demonstrate how compound reward functions, reward functions to generate sequences of tasks, can be created that allow the mobile robot to learn more complex behaviors.

Keywords: intrinsic reward function, goal types, open-ended learning, autonomous goal generation, reinforcement learning

Edited by:
Vieri Giuliano Santucci, Istituto di Scienze e Tecnologie della Cognizione (ISTC), Italy

Reviewed by:
Gianluca Baldassarre, Consiglio Nazionale Delle Ricerche (CNR), Italy
Eiji Uchibe, Advanced Telecommunications Research Institute International (ATR), Japan
Patricia Shaw, Aberystwyth University, United Kingdom

***Correspondence:**
Paresh Dhakan
dhakan-p@ulster.ac.uk

Received: *30 April 2018*
Accepted: *11 September 2018*
Published: *09 October 2018*

Citation:
Dhakan P, Merrick K, Rañó I and Siddique N (2018) Intrinsic Rewards for Maintenance, Approach, Avoidance, and Achievement Goal Types. Front. Neurorobot. 12:63. doi: 10.3389/fnbot.2018.00063

INTRODUCTION

Open-ended learning, still an open research problem in robotics, is envisaged to provide learning autonomy to robots such that they will require minimal human intervention to learn environment specific skills. Several autonomous learning frameworks exist (Bonarini et al., 2006; Baranes and Oudeyer, 2010a,b; Santucci et al., 2010, 2016), most of which have similar key modules that include: (a) a goal generation mechanism that discovers the goals the robot can aim to achieve; and (b) a learning algorithm that enables the robot to generate the skills required to achieve the goals. Many of the autonomous learning frameworks use reinforcement learning (RL) as the learning module (Bonarini et al., 2006; Santucci et al., 2010, 2016). In RL, an agent learns by trial and error. It is not initially instructed which action it should take in a particular state but instead must compute the most favorable action using the reward as feedback on its actions. For many dynamic environments, however, it is not always possible to know upfront which tasks the agent should learn. Hence, sometimes, it is not possible to design the reward function in advance. Open-ended learning aims to build systems that autonomously learn tasks as acquired skills that can later be used to learn user-defined tasks more efficiently (Thrun and Mitchell, 1995; Weng et al., 2001; Baldassarre and Mirolli, 2013). Thus, for an open-ended learning system, autonomous reward function generation is an essential component. This paper contributes to open-ended learning by proposing an approach to reward function generation based on the building blocks of maintenance, achievement, approach and avoidance goals.

Existing literature reveals two common solutions to address the problem of the autonomous reward function design or at least provides a level of autonomy in designing a reward function: (1) Intrinsic motivation (Singh et al., 2005) and (2) reward shaping (Ng et al., 1999; Laud and DeJong, 2002). Intrinsic motivation is a concept borrowed from the field of psychology. It can be used to model reward that can lead to the emergence of task-oriented performance, without making strong assumptions about which specific tasks will be learned prior to the interaction with the environment. Reward shaping, on the other hand, provides a positive or negative bias encouraging the learning process toward certain behaviors. Intrinsic motivation, although promising, has not been validated on large-scale real-world applications, and reward shaping requires a significant amount of domain knowledge thus cannot be considered as an autonomous approach. As an alternative to these solutions, we propose reward functions based on the various types of goals identified in the literature. Although the concept of creating a reward function using goals is not new, this approach is often overlooked and has not been the main focus of the RL community. In our approach, different reward functions are generated based on the type of the goal, and since the reward functions exploit the inherent property of each type of goal, these reward functions are task-independent.

Goals have been the subject of much research within the Beliefs, Desires, Intentions community (Rao and Georgeff, 1995), and the agent community (Regev and Wegmann, 2005). A goal is defined as an objective that a system should achieve (Van Lamsweerde, 2001), put another way, a goal is the state of affairs a plan of action is designed to achieve. Goals range in abstraction from high-level to low-level, cover functional as well as non-functional aspects and can be categorized into hard goals that can be verified in a clear-cut way to soft goals that are difficult to verify (Van Lamsweerde, 2001). Examples of types of goals include achievement, maintenance, avoidance, approach, optimization, test, query, and cease goals (Braubach et al., 2005). Instead of classifying goals based on types, Van Riemsdijk et al. (2008) classify them as declarative or state-based where the goal is to reach specific desired situation and procedural or action-based where the goal is to execute actions. State-based goals are then sub-classified into the query, achieve and maintain goals, and action-based goals are sub-classified into perform goal. RL is already able to solve some problems where some of these kinds of goals are present. For example, well-known benchmark problems such as the cart-pole problem are maintenance goals, while others such as maze navigation are achievement goals. Likewise, problems solved with positive reward have typically approach goal properties, while problems solved from negative reward have avoidance goal properties. The idea of generating reward signals for generic forms of these goals thus seems promising. Based on this logic we propose a domain-independent reward function for each of the goal types. This approach can be applied to the goal irrespective of its origin, i.e., whether the goal is intrinsic, extrinsic or of a social origin. In this paper though, we use the output of an existing goal generation module for a mobile robot (Merrick et al., 2016) to validate the proposed reward functions. We show how the intrinsic reward functions bridge the gap between goal generation and learning by providing a task-independent reward. We further demonstrate how these primitive reward functions based on the goal types can be combined to form compound reward functions that can be used to learn more complex behaviors in agents. Thus, the contributions of this paper are: (1) A proposal for task-independent intrinsic reward functions for maintenance, approach, avoidance and achievement goal types; (2) Metrics for the measurement of the performance of these reward functions with respect to how effectively solutions to them can be learned; and (3) A demonstration of how these primitive reward functions can be combined to motivate learning of more complex behaviors.

The remainder of the paper is organized as follows. In section Background and Related Work, we present a background on the design of reward functions and the solutions for task-independent reward functions found in the literature. In section Primitive Goal-based Motivated Reward Functions, we detail the proposed reward functions based on the goal types, and the metrics we use to measure the agents' performance using those reward functions. In section Experiments for Maintenance, Approach, Avoidance and Achievement Goal Types, we detail experiments to examine the performance of reward functions for maintenance, approach, avoidance, and achievement goal types on a mobile "e-puck" robot. In section Demonstration of how Primitive Goal-based Reward Functions can be Combined, we demonstrate complex behaviors learned from compound reward functions constructed from the autonomously generated primitive functions for each goal type. Finally, in section Conclusion and Future Work, we provide concluding remarks and discuss directions for future work.

BACKGROUND AND RELATED WORK

In RL, an agent perceives the state of its environment with its sensors and takes action to change that state. The environment may comprise variables such as the robot's position, velocity, sensor values, etc. These parameters collectively form the state of the agent. With every action that the agent executes in the environment, it moves to a new state. The state of the agent at time t can be expressed as:

$$S_t = \left[s_t^1,\ s_t^2, s_t^3, \ldots, s_t^n\right]$$

where each attribute s_t^i is typically a numerical value describing some internal or external variable of the robot, and n is the number of attributes of the state. The agent takes an action A_t to change the state of the environment from the finite set of m actions:

$$A = \{\ A^1,\ A^2, A^3, \ldots, A^m\}$$

This state change is denoted by event E_t, formally denoted as:

$$E_t = \left[e_t^1,\ e_t^2, e_t^3, \ldots, e_t^n\right]$$

where an event attribute $e_t^i = s_t^i - s_{t-1}^i$. That is,

$$E_t = S_t - S_{t-1} = \left[\Delta(s_t^1 - s_{t-1}^1),\ \Delta(s_t^2 - s_{t-1}^2), \ldots, \Delta(s_t^n - s_{t-1}^n)\right]$$

Thus, an event, which is a vector of difference variables, models the transition between the states. An action can cause a number of different transitions, and an event is used to represent those transitions. Since this representation does not make any task-specific assumption about the values of the event attributes, it can be used to represent the transition in a task-independent manner (Merrick, 2007).

Finally, the experience of the agent includes the states S_t it has encountered, the events E_t that have occurred and the actions A_t that it has performed. Thus, the experience X is a trajectory denoted as the following, and it provides the data from which the goals can be constructed.

$$X = \{S_0, A_0, S_1, E_1, A_1, S_2, E_2, A_2, S_3, E_3, \ldots\}$$

Design of Reward Functions

In RL, the reward is used to direct the learning process. A simple example of a reward function is a pre-defined value assignment for known states or transitions. For example:

$$r(S_t) = \begin{cases} 1 & \text{if a paricular state } S_t \text{ is reached} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

A more specific, task-dependent example can be seen from the canonical cart-pole domain in which a pole is attached to a cart that moves along a frictionless track. The aim of the agent is to maintain the pole balanced on the cart by moving the cart to the right or left. The reward, in this case, depends on the attributes specific to the task:

$$r(S_t) = -c2^* \left(G^1 - s_t^1\right)^2 - c3^*(G^2 - s_t^2)^2 \tag{2}$$

where s_t^1 is the position of the cart and s_t^2 is the angle of the pole with respect to the cart, G (with attributes G^1–desired position and G^2–desired angle) is the goal state, and $c2$ and $c3$ are constants.

For an even more complex task like ball paddling, where a table-tennis ball is attached to a paddle by an elastic string with the goal to bounce the ball above the paddle, it is quite difficult to design a reward function. Should the agent be rewarded for bouncing the ball a maximum number of times? Should the agent be rewarded for keeping the ball above the paddle? As detailed in Amodei et al. (2016), the agent might find ways to "hack the reward" resulting in unpredictable or unexpected behavior.

For some complex domains, it is only feasible to design "sparse reward signals" which assign non-zero reward in only a small proportion of circumstances. This makes learning difficult as the agent gets very little information about what actions resulted in the correct solution. Proposed alternatives for such environments include "hallucinating" positive rewards (Andrychowicz et al., 2017) or bootstrap with self-supervised learning to build a good world model. Also, imitation learning and inverse RL have shown reward functions can be implicitly defined by human demonstrations, so they do not allow a fully autonomous development of the agent.

"Reward engineering" is another area that has attracted the attention of the RL community, which is concerned with the principles of constructing reward signals that enable efficient learning (Dewey, 2014). Dewey (2014) concluded that as artificial intelligence becomes more general and autonomous, the design of reward mechanisms that result in desired behaviors are becoming more complex. Early artificial intelligence research tended to ignore reward design altogether and focused on the problem of efficient learning of an arbitrary given goal. However, it is now acknowledged that reward design can enable or limit autonomy, and there is a need for reward functions that can motivate more open-ended learning beyond a single, fixed task. The following sections review work that focus in this area.

Intrinsic Motivation

Reward modeled as intrinsic motivation is an example of an engineered reward leading to open-ended learning (Baldassarre and Mirolli, 2013). It may be computed online as a function of experienced states, actions or events and is independent of *a priori* knowledge of task-specific factors that will be present in the environment. The signal may serve to drive acquisition of knowledge or a skill that is not immediately useful but could be useful later on (Singh et al., 2005). This signal may be generated by an agent because a task is inherently "interesting," leading to further exploration of its environment, manipulation/play or learning of the skill.

Intrinsic motivation can be used to model reward that can lead to the emergence of task-oriented performance, without making strong assumptions about which specific tasks will be learned prior to the interaction with the environment. The motivation signal may be used in addition to a task-specific reward signal, aggregated based on a predefined formula, to achieve more adaptive, and multitask learning. It can also be used in the absence of a task-specific reward signal to reduce the handcrafting and tuning of the task-specific reward thus moving a step closer to creating a true task independent learner (Merrick and Maher, 2009). Oudeyer and Kaplan (2007) proposed the following categories for a computational model of motivation: knowledge-based, and competence-based. In knowledge-based motivation, the motivation signal is based on an internal prediction error between the agent's prediction of what is supposed to happen and what actually happens when the agent executes a particular action. In competence-based motivation, the motivation signal is generated based on the appropriate level of learning challenge. This competency motivation depends on the task or the goal to accomplish. The activity at a correct level of learnability given the agent's current level of mastery of that skill generates maximum motivation signal. Barto et al. (2013) further differentiated between surprise (prediction error) and novelty based motivation. Novelty motivation signal is computed based on the experience of an event that was not experienced before (Neto and Nehmzow, 2004; Nehmzow et al., 2013).

Intrinsically Motivated Reinforcement Learning

Frameworks that combine intrinsic motivation with RL are capable of autonomous learning, and they are commonly termed intrinsically motivated reinforcement learning frameworks.

Singh et al. (2005) and Oudeyer et al. (2007) state that intrinsic motivation is essential to create machines capable of lifelong learning in a task-independent manner as it favors the development of competence and reduces reliance on externally directed goals driving learning. When intrinsic motivation is combined with RL, it creates a mechanism whereby the system designer is no longer required to program a task-specific reward (Singh et al., 2005). An intrinsically motivated reinforcement learning agent can autonomously select a task to learn and interact with the environment to learn the task. It results in the development of an autonomous entity capable of resolving a wide variety of activities, as compared to an agent capable of resolving only a specific activity for which a task-specific reward is provided.

Like in RL, in an intrinsically motivated reinforcement learning framework, the agent senses the states, takes actions and receives an external reward from the environment, however as an additional element, the agent internally generates a motivation signal that forms the basis for its actions. This internal signal is independent of task-specific factors in the environment. Incorporating intrinsic motivation with RL enables agents to select which skills they will learn and to shift their attention to learn different skills as required (Merrick, 2012). Broadly speaking, intrinsically motivated reinforcement learning introduces a meta-learning layer in which a motivation function provides the learning algorithm with a motivation signal to focus the learning (Singh et al., 2005).

Role of Goals to Direct the Learning

Where early work focused on generating reward directly from environmental stimuli, more recent works have acknowledged the advantages of using the intermediate concept of a goal to motivate complexity and diversity of behavior (Merrick et al., 2016; Santucci et al., 2016). It has been shown by Santucci et al. (2012) that using intrinsic motivation (generated by prediction error) directly for skill acquisition can be problematic and a possible solution to that is to instead generate goals using the intrinsic motivation which in turn can be used to direct the learning. Further, it has been argued by Mirolli and Baldassarre (2013) that a cumulative acquisition of skills requires a hierarchical structure, in which multiple "expert" sub-structures focus on acquiring different skills and a "selector" sub-structure decides which expert to select. The expert substructure can be implemented using knowledge-based intrinsic motivation that decides what to learn (by forming goals), and the selector sub-structure can be implemented using competence-based intrinsic motivation that can be used to decide which skill to focus on. Goal-directed learning is also shown to be a promising direction for learning motor skills. Rolf et al. (2010) show how their system auto-generates goals using inconsistencies during exploration to learn inverse kinematics and that the approach can scale for a high dimension problem.

Recently, using goals to direct the learning has even attracted the attention of the deep learning community. Andrychowicz et al. (2017) have proposed using auto-generated interim goals to make learning possible even when the rewards are sparse. These interim goals are used to train the deep learning network using experience replay. It is shown that the RL agent is able to learn to achieve the end goal even if it has never been observed during the training of the network. Similarly, in a framework proposed by Held et al. (2017), they auto-generate interim tasks/goals at an appropriate level of difficulty. This curriculum of tasks then directs the learning enabling the agent to learn a wide set of skills without any prior knowledge of its environment.

Regardless of whether the goals are intrinsic, extrinsic, of social origin, whether they are created to direct the learning or generated by an autonomous learning framework, the approach of using goal-based reward functions detailed in the next section can be applied to them.

PRIMITIVE GOAL-BASED MOTIVATED REWARD FUNCTIONS

The basis of our approach in this paper is a generic view of the function in Equation (1) as follows:

$$r(S_t) = \begin{cases} 1 & \text{if the goal is reached} \\ 1 - \varepsilon & \text{otherwise} \end{cases} \quad (3)$$

where ε is a non-negative constant. The remainder of this section defines different representations of "goal" in Equation (3) and representation of the meaning of "reached."

Reward Function for the Maintenance Goal Type

A maintenance goal monitors the environment for some desired world state and motivates the agent to actively try to re-establish that state if the distance between the desired state and the current state goes beyond a set limit. For a maintenance goal, an agent's action selection should consider both triggering conditions as well as the constraining nature of the goal (Hindriks and Van Riemsdijk, 2007). The act of maintaining a goal can be never-ending thus making the process continuous or non-episodic.

Consider G is the state that the agent desires to maintain. The state is considered as maintained if the distance between the current state and desired state is sufficiently small. The reward at time step t can then be expressed as:

$$r(S_t) = \begin{cases} \sigma & \text{if } d(S_t, G) < \rho \\ \varphi & \text{otherwise} \end{cases} \quad (4)$$

where $d(.)$ is a distance function, S_t is the current state, G is the desired goal state and ρ is a defined distance threshold. The reward for when the goal is maintained is σ and the reward for other time steps is φ, with $\varphi < \sigma$ in order to incentivize the agent to find a shorter path to reach the goal state. σ is generally 0 or a positive number to provide positive reinforcement.

We hypothesize that there are various ways in which an agent's performance can be measured with respect to a maintenance goal. For example, the following metrics M evaluate the reward function for the maintenance goal type. Each metric is assumed to be measured over a fixed period T of the agent's life.

- **Number of Steps for Which the Goal is Maintained (M1).** This metric counts the total number of times the agent maintains the state for two or more consecutive steps during a period T. Note that since the process of maintaining a goal is continuous, we do not assume the end of a learning episode at the first occurrence of the goal being maintained. For such non-episodic processes, there may be a reason why the maintained state is lost. Thus, this metric provides the measurement of the agent's ability to learn to regain the maintenance goal.

$$M_1 = \underset{t=2\ldots T}{\text{count(t)}}\ \textit{such that}\ \ r_t = 1\ \textit{and}\ r_{t-1} = 1$$

- **Number of Steps the Goal is Accomplished (M_2).** This metric provides an alternative to M_1 and counts the total number of steps for which the agent receives a positive reward. This metric provides the measurement of the number of time steps the agent managed to maintain the goal. A higher value indicates ease of maintainability of a particular goal.

$$M_2 = \underset{t=1\ldots T}{\text{count(t)}}\ \textit{such that}\ \ r_t = 1$$

- **Average Number of Steps of Consecutive Goal Maintenance (M_3).** This measures the length of time (on average) that positive consecutive positive reward is received. This metric also provides an indication of the ease of maintainability of a particular goal. It is calculated by first calculating how many times J a goal was reacquired (that is, how many times $r_t = 1\ \textit{and}\ r_{t-1} \neq 1$) and dividing M_2 as follows:

$$M_3 = \frac{M_2}{J}$$

- **Longest Period of Goal Maintenance (M_4).** This metric finds the length of the longest stretch for which the agent was able to maintain the goal. This metric indicates the final ability accomplished by the agent in maintaining the goal. Longer stretches indicate better progress in learning to maintain the desired goal state.

$$M_4 = \max_{j=1\ldots J}(\textit{length of maintenance period } j)$$

Reward Function for the Approach Goal Type

An approach goal represents the agent's act of attempting to get closer to the desired world state. The main difference between an approach and maintenance goal lies in the condition of fulfillment. An approach goal is fulfilled when the agent is getting closer to the desired state whereas a maintenance state is fulfilled when the desired state is maintained and not violated. An approach attempt leads to a behavior that functions to shorten the distance, either physically or psychologically between the agent and the desired outcome (Elliot, 2008).

The reward function for the approach goal can be expressed as:

$$r(S_t) = \begin{cases} \sigma & \text{if } d\,(S_t,\ G) < d\,(S_{t-1},\ G)\ \text{ and }\ d\,(S_t,\ G) > \rho \\ \varphi & \text{otherwise} \end{cases} \tag{5}$$

where *d(.)*, the distance function is used to check the approach attempt by comparing the distance between the current state S_t and the desired goal state G with the distance between the previous state S_{t-1} and G. The second condition of the equation ensures that the distance is more than the defined distance threshold ρ so that "reached" means an approach attempt and not "approach and achieve". Same as in Equation (4), the reward for when the goal is not reached is φ with $\varphi < \sigma$ in order to incentivize the agent to find a shorter path to the goal state.

The following metrics may thus be used to evaluate this reward function for the approach goal type. Each metric is again assumed to be measured over a fixed period T of the agent's life. Since the approach and avoidance functions (detailed in section Reward Function for the Avoidance Goal Type) reward the approach and the avoidance attempts irrespective of the distance between the current and the goal state, the cumulative reward for the agent is very high. In order to get a better sense of the proportion of the reward gained per trial, we use percentage in the following metrics.

- **Number of Steps the Goal is Approached as a Percentage of *T* (M_5).** This metric indicates the approachability of the goal, i.e., how easy is it to approach the goal state?

$$M_5 = \frac{M_2 \times 100}{T}$$

- **Number of Approach Attempts as a Percentage of *T* (M_6).** The agent is considered to have made an approach attempt if it receives a positive reward for two or more consecutive steps, i.e., signifying that the agent attempted to approach the goal state.

$$M_6 = \frac{M_1 \times 100}{T}$$

Reward Function for the Avoidance Goal Type

An avoidance goal type is the opposite of the approach goal type. Avoidance is a behavior where an agent stays away or moves away from an undesirable stimulus, object or event (Elliot, 2008). An avoidance goal is considered fulfilled as long as the agent is away from the state that it wants to avoid, and it increases the distance from the state that it wants to avoid. Considering those definitions, the reward function for avoidance goal has two expressions, one that fulfills the condition of moving away from the goal state and other that fulfills the condition of staying away from the goal state, however, in the applications either of the other expressions can be used on their own.

$$r(S_t) = \begin{cases} \sigma & \text{if } d\,(S_t,\ G) > d\,(S_{t-1},\ G)\ \text{ and }\ d\,(S_{t-1},\ G) > \rho \\ \varphi & \text{otherwise} \end{cases} \tag{6}$$

Similar to Equation (5), there are two conditions in Equation (6). The first condition checks for the avoidance attempt, while the second checks that the distance between the previous state S_{t-1} and the desired goal state G is above the defined distance threshold ρ, i.e., the current state is not G. Same as in Equation (4), the reward for when the goal is not avoided is φ with $\varphi < \sigma$.

Both the metrics M_5 and M_6 are applicable to avoidance goals. In addition, it is also possible to measure:

- **Number of Times Goal Not Avoided *(M_7)*.** This is a count of a number of times the agent fails to avoid the goal state.

$$M_7 = \underset{t=1\ldots T}{\text{count(t)}} \; \textit{such that} \;\; d\,(S_t,\ G) < \rho$$

Reward Function for the Achievement Goal Type

An achievement goal is a state of the world that the agent strives to fulfill (Duff et al., 2006), i.e., the state that the agent wants to bring about in the future. When this target state is reached, the goal is considered as succeeded. The learning process can be restarted with a same/different initial starting state making the process episodic if required.

Similar to Merrick et al. (2016), we use the concept of an event detailed in section Background and Related Work to represent an achievement task. An event (given as $E_t = S_t - S_{t-1}$) allows the agent to represent a change in its environment. An achievement goal defines changes in the event attributes e_t^i that the agent should bring about. Thus, the reward for the achievement goal can be generated in response to the experience of event E_t as:

$$r(S_t, S_{t-1}) = \begin{cases} \sigma & \text{if } d\,(E_t,\ G) < \rho \\ \varphi & \text{otherwise} \end{cases} \qquad (7)$$

where similar to Equations (4–6), ρ is the distance threshold, σ is generally 0 or a positive number to provide positive reinforcement and $\varphi < \sigma$ in order to incentivize the agent to find a shorter path to reach the goal state. The metric M_2 is most useful for measuring the performance of this goal type.

The next section uses the metrics proposed in this section to evaluate the goal-based reward functions detailed by Equations (4)–(7).

EXPERIMENTS FOR MAINTENANCE, APPROACH, AVOIDANCE, AND ACHIEVEMENT GOAL TYPES

We used Webots software to simulate an e-puck mobile robot. E-puck is a small differential wheeled mobile robot with eight proximity sensors, of which we used 6. The sensors are labeled in a clockwise direction as *Front-Right, Right, Rear-Right, Rear-Left, Left, and Front-Left*. The red lines in **Figure 1A** show the direction in which the sensors detect an obstacle. A high sensor reading indicates that an object is close to that sensor. **Figure 1B** shows a 5 × 5 m square flat walled arena that we use for our experimentation with primitive goal-based reward functions.

The arena, the state, and the action space of the robot are the same as detailed by Merrick et al. (2016). The state of the mobile robot comprises nine parameters: left wheel speed, right wheel speed, orientation, left sensor value, right sensor value, front-left sensor value, front-right sensor value, rear-left sensor value, and rear right sensor value, i.e., the state vector is [$\omega^R\ \omega^L\ \theta\ s^L\ s^R\ s^{FL}\ s^{FR}\ s^{RL}\ s^{RR}$]. ω^R and ω^L are the rotational velocities of the right and the left wheels. Their range is −π to π radians per second. θ is the orientation angle of the mobile robot. Its value ranges from −π to π. For our experiments, we use binary values for the proximity sensors with 0 indicating that there is no object in the proximity of the sensor, and 1 indicates that the object is near. The rotational velocities and orientation are discretized into nine values making the state space quite large.

The action space comprises five actions: 1–increase the left wheel speed by δ, 2–increase the right wheel speed by δ, 3–decrease the left wheel speed by δ, 4–decrease the right wheel speed by δ, and 5–no change to any of the wheel speeds. A fixed value of π/2 was used as δ.

In this paper, we use the goals generated for the mobile robot based experiment by Merrick et al. (2016). The main concept of the experience based goal generation detailed in Merrick et al. (2016) is that the agent must explore its environment and determine if the experience is novel enough to be termed a potential goal. Goal generation phase is divided into two stages: experience gathering stage and the goal clustering stage. In the experience gathering stage, the mobile robot moves around randomly in its environment. The states experienced by the robot are recorded. These recorded states form an input to the goal clustering stage which uses simplified adaptive resonance theory (SART) network (Baraldi, 1998). SART is a neural network based clustering technique. It is capable of handling a continuous stream of data thus solving the stability-plasticity dilemma. The network layer takes a vector input and identifies its best match in the network. Initially, the network starts with no clusters. As the data is read, its similarity is checked with any existing clusters. If there is close enough match, it is clustered together else a new cluster is created. As the clusters are created, they are connected to the input nodes (i.e., the recorded experience). The number of clusters created will depend on the vigilance parameter of the SART network. Higher vigilance produces many fine-grained clusters whereas a low vigilance parameter produces a coarser level of clusters. The goals generated by this phase form input for the goal learning phase.

In the learning phase, the robot learns the skills to accomplish the goals. For the goal learning, we use an RL algorithm called Dyna-Q. Dyna-Q (Sutton and Barto, 1998) is a combination of Dyna architecture with RL's Q Learning algorithm. With Dyna-Q, the Q-Learning is augmented with model learning, thus combining both model-based and model-free learning. The RL agent improves its Q value function using both the real experiences with its environment and imaginary experiences (also called planning process) generated by the model of the environment. During the planning process, that is typically run several times for every real interaction with the environment; the algorithm randomly selects the samples from the model (continuously updated using the real experiences) and updates the Q value function. This reduces the number of interactions required with the environment which are typically expensive, especially for robotic applications. The model of the environment for our experiments keeps track of the state *s'* that the mobile robot lands in when it takes a particular action *a* in the current state *s*. The model also keeps track of the reward that the robot receives during that transition. The state transitions for our

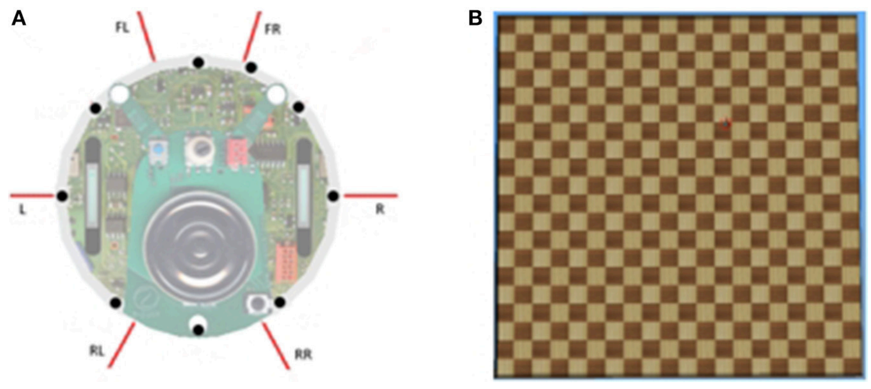


FIGURE 1 | (A) e-puck proximity sensors (shown by the red directional lines). **(B)** A sample walled arena.

experiments are deterministic in nature, i.e., when the robot takes action *a* in state *s*, it will always land in a state *s'*. The number of iterations for model learning can be varied as required. We set this parameter to 25, i.e., the algorithm will attempt 25 actions for model learning (using imaginary experiences) before attempting one action with the real environment.

Maintenance Goal Learning Results

Merrick et al. (2016) used SART based clustering to generate two sets of goals, namely, maintenance and achievement goals. **Table 1** lists the set of maintenance goals described by the ID, goal attributes and the meaning of the goal as detailed by Merrick et al. (2016). These goals are the actual states experienced by the mobile robot. This same set of goals are used in section Approach Goal Results and Avoidance Goal Results treated as approach and avoidance type, respectively. **Table 4** in section Achievement Goals, lists the set of achievement goals generated by Merrick et al. (2016).

Table 1 also shows the results of the experiments for these goals treated as maintenance goals. The columns M_1, M_2, M_3, and M_4 are the metrics detailed in section Reward Function for the Maintenance Goal Type. The goals are states experienced by the mobile robot treated as maintenance goal for these experiments, i.e., the aim of the robot is to maintain these goal states. The e-puck mobile robot simulation was run for 10 trials each of 25,000 steps for each of the 12 goals. Results were averaged over 10 trials, and the standard deviation is also shown in the table. Values of the parameters of Equation (4) were as follows: ρ was 0.9, σ was 1, φ was -1 and *d* was the Euclidian distance. The RL exploration parameter epsilon was set to 0.15, and the decay schedule was linear. When a trial ended, the end position and orientation of the e-puck mobile robot became the start position and orientation for the next trial. However, the RL Q table was reset after each trial, so no learning was carried forward between the trials.

Once the robot reaches the goal state, it maintains it until it comes across adverse conditions, i.e., for G^1 (move forward at high speed), once the goal state is reached, the robot will maintain that state while it is in the open space. However, once it reaches a wall, it is not able to maintain the state. We consider that the robot has learnt to attain the goal if the robot is able to reach the goal state over and over again and remain in that state for two time-steps or more. This is indicated by the column for M_1. This measure is high for G^1, G^2, G^3, G^4, G^6, G^7, G^{10}, and G^{12} indicating that the robot is able to maintain those goals. However, that measure is very low for goal G^5 and zero for G^8 which shows that the robot is not able to learn to maintain those goal states. This is due to the lack of opportunity, i.e., the robot has to be in a specific situation to be able to learn to maintain those goals. Those goals require the robot to be close to a wall, the likelihood of which is small because of the size of the arena.

The measure M_1 for goal G^9, which is a valid goal, is zero. The mobile robot was not able to achieve that goal because of the lack of opportunity. The required situation to learn that goal would be that the robot should find itself in the bottom left corner at a particular orientation. The measure M_1 is zero for G^{11} as well. The reason for that is because goal G^{11} is an unreasonable goal. According to that state, the wall is close to the Right and Front Left sensors but not Front Right. It is hard to imagine a position of the mobile robot that represents such state. The goals created by SART are the cluster centers. It appears that this is an example of the clustering algorithm creating a hybrid, unreasonable goal which could be either because the granularity of the clusters is coarser than it should be, resulting in the cluster centroid not being a correct representative of the cluster or that invalid states experienced by the robot due to noise. The column "Is Goal Valid?" is marked "No" in this case.

Figure 2A shows a sample trajectory of the mobile robot for G^1. The trajectory is a two-dimensional plot of the path followed by the mobile robot in the arena during the trial. The goal is attained by maintaining a high speed at a particular orientation. The robot receives a positive reward for the time steps that it maintains the goal. It is only possible for the robot to attain G^1 when it is in the open area of the arena. When it reaches the wall, it is no longer able to maintain goal G^1. The robot has to learn to turn around and attain the goal again. This is evident in **Figure 2A** that shows multiple straight stretches where the

Table 1 | Experiments and results for maintenance goals.

ID	Goal attributes	Meaning of the goal	M_1	M_2	M_3	M_4	Is goal valid?
G^1	(2.5, 2.5, 1.8, 0, 0, 0, 0, 0, 0)	Move forward at high speed	37 ± 8	493 ± 91	14 ± 4	154 ± 7	Yes
G^2	(0.4, 0.4, 1.2, 0, 0, 0, 0, 0, 0)	Move forward at low speed	121 ± 25	568 ± 124	4 ± 1	88 ± 0	Yes
G^3	(−2.4, −2.4, 1.4, 0, 0, 0, 0, 0, 0)	Move backward at high speed	88 ± 8	888 ± 179	10 ± 2	188 ± 9	Yes
G^4	(−0.4, −0.4, −1.3, 0, 0, 0, 0, 0, 0)	Move backward at low speed	192 ± 28	866 ± 110	4 ± 0	71 ± 0	Yes
G^5	(0.0, 0.0, −2.8, 0, 1, 0, 0, 0, 0)	Stop for obstacle in front	1 ± 1	3 ± 3	1 ± 0	5 ± 0	Yes
G^6	(−0.4, −0.4, 2.9, 0, 0, 0, 0, 0, 0)	Move backward at low speed	142 ± 24	601 ± 106	4 ± 0	37 ± 1	Yes
G^7	(−0.8, −0.8, 1.6, 0, 0, 0, 0, 0, 0)	Move backward at moderate speed	157 ± 26	848 ± 127	5 ± 0	53 ± 2	Yes
G^8	(0.2, 0.0, 2.4, 1, 0, 0, 0, 0, 1)	Stop for obstacle behind	0 ± 0	0 ± 0	0 ± 0	0 ± 0	Yes
G^9	(0.0, −0.3, 2.1, 1, 0, 0, 0, 1, 0)	Stop for obstacle at left and back	0 ± 0	0 ± 0	0 ± 0	2 ± 0	Yes
G^{10}	(−1.9, −1.9, −2.2, 0, 0, 0, 0, 0, 0)	Move backward at moderate speed	162 ± 23	763 ± 105	4 ± 0	52 ± 2	Yes
G^{11}	(0.0, 0.0, 3.0, 0, 1, 1, 0, 0, 0)	Stop for obstacle in front	0 ± 0	0 ± 0	0 ± 0	0 ± 0	No
G^{12}	(1.2, 1.2, −2.7, 0, 0, 0, 0, 0, 0)	Move forward at moderate speed	100 ± 18	427 ± 85	4 ± 0	36 ± 1	Yes

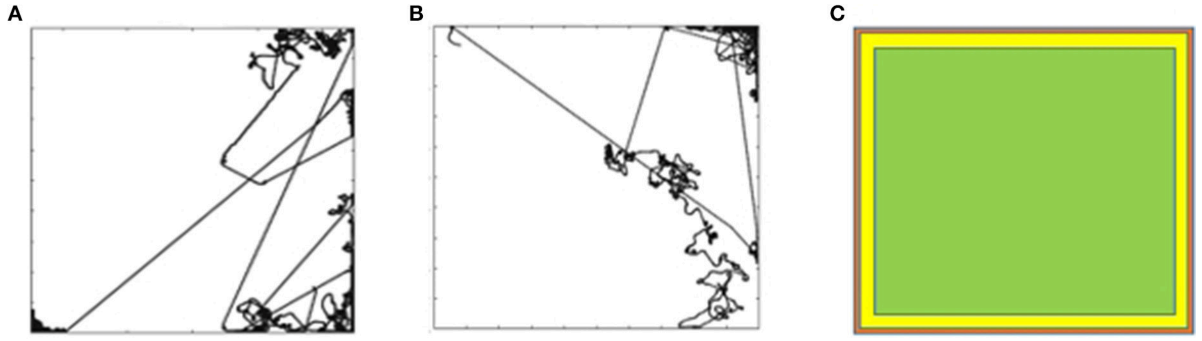


FIGURE 2 | (A) Mobile robot trajectory for G1. **(B)** Mobile robot trajectory for G3. **(C)** Likelihood of the reward for G1, G3, and G12.

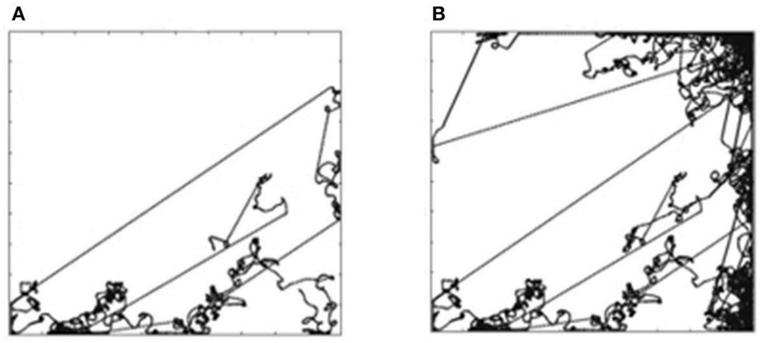


FIGURE 3 | (A) Trajectory for G12. **(B)** Simultation for a G12 run for 100,000 steps.

robot attains G^1, reaches the wall, tries to turn around and attains the goal again. **Figure 2B** shows the trajectory of the mobile robot for G^3 (move backward at high speed) and **Figure 3A** shows the trajectory for goal G^{12} (move forward at moderate speed).

For goals G^1, G^3, and G^{12} the robot is only able to attain the goals when it is in the open area of the arena. **Figure 2C** shows the likelihood diagram with the wall shown in orange. In the open area of the arena shown in green, the robot is more likely to attain the goal, i.e., to receive a positive reward. In the area close to the wall (shown in yellow) the likelihood reduces. The probability of the mobile robot to be in the green zone can be calculated as follows for the environment with the size of the board 5×5 m and sensor range of e-puck 0.06 m. If we were to discretize the environment into squares of 0.06 m, then there would be 83×83, i.e., 6,889 squares in the grid. Green zone for G^1, G^3, and G^{12} will consist of 81×81, i.e., 6,561 squares. If we were to randomly select a square in the green zone, the probability would be (81

$\times$ 81)/(83 $\times$ 83) = 95.23%. The orientation and wheel speeds are divided into nine buckets each. Hence the probability of the robot to be in a particular square with particular wheel speed and orientation will be (81 $\times$ 81)/(83 $\times$ 83 $\times$ 9 $\times$ 9 $\times$ 9) = 0.13%.

For G^{12} we let the simulation for one of the trials continue for 100,000 steps, the trajectory of which is shown in **Figure 3B**. The straight-line trajectory shows that the robot is maintaining the goal of moving forward at a moderate speed, i.e., it is in the region of opportunity (**Figure 2C**). When the robot reaches the wall, it experiences states that it may not have experienced in the past. However, it eventually learns to attain the goal of moving forward at a moderate speed.

Figures 4A,B shows the trajectory for goal G^5 (stop for an obstacle in front) and G^8 (stop for obstacle behind), respectively. The robot does not learn to attain these goals. The obstacles in the arena are the four walls hence the likelihood of the reward are the areas closer to the wall. Considering the orientation for goals G^5 and G^8, the mobile robot has to be beside the top wall as shown in green in **Figure 4C**. The probability of the mobile robot to be in a particular square with the orientation required for G^5 or G^8 is (81)/(83 $\times$ 83 $\times$ 9 $\times$ 9 $\times$ 9) = 0.002%. This lack of opportunity is the reason why the robot does not learn G^5 and G^8 goals. In order to confirm this hypothesis, we continued the experiments with these two goals with the reduced arena size. The size of the arena was reduced to 0.25 $\times$ 0.25 m to increase the opportunity for the mobile robot to be near a wall. In that arena, the probability of the mobile robot to find itself in the required situation is increased by the factor of 400 (20 $\times$ 20) to 0.65%, thus increasing its ability to attain G^5 and G^8 goals.

Approach Goal Results

Table 2 shows the results of the experiments for the approach goals. The 12 goals and their corresponding goal IDs, goal attributes and the meaning of the goal, are the same as the goals detailed in **Table 1**. The goals for these set of experiments will be treated as approach goals, i.e., the aim of the robot is to approach those goal states. Values of the parameters of Equation (5) and the method in which experiments were conducted for the approach goals were the same as detailed in section Maintenance Goal Learning Results.

The design of the reward function for the approach goal type is such that it rewards an approach attempt. Hence if the agent is getting closer to the goal, it receives a positive reward. Goals, when treated as approach goals, are relatively straightforward to attain as seen in the M_5 column in **Table 2** (average number of steps positive reward received as a percentage). In the case of the goal G^1, for instance, the agent receives a positive reward for 32.49% of the time steps. This is because the attempt to approach the goal is rewarded irrespective of the distance between the current state and the goal state. Results also show that all the goals, when treated as approach type, are attainable (even the invalid goals) indicating that it is possible to approach the goal states of each of the 12 goals.

Avoidance Goal Results

Table 3 shows the results of the experiments for the avoidance goals. The 12 goals and their corresponding goal IDs, goal attributes, and the meaning of the goal, are the same as the goals detailed in **Table 1**. The goal states for these experiments are treated as avoidance goals, i.e., the aim of the robot is to avoid those goal states. Values of the parameters of Equation (6) and the method in which experiments were conducted for the avoidance goals were the same as detailed in section Maintenance Goal Learning Results.

The reward function for the avoidance goal type rewards the attempt to avoid the goal, i.e., the agent is moving away from the goal state. As it can be seen in the table, the goals, when treated as avoidance goals, are relatively easy to attain. This is because the attempt to avoid the desired goal state is rewarded irrespective of the distance between the current state and the goal state. Based on the M_7 column (average number of times the goal state was not avoided), it can be said that even the goals that are difficult to attain due to lack of opportunity, when treated as maintenance goals (for example, G^5, G^8, and G^9), are easier to avoid when treated as avoidance goals.

Achievement Goal Results

Table 4 lists the set of achievement goals generated by Merrick et al. (2016). The goal ID, goal attributes, and the meaning of the goal are the output of the SART based clustering as detailed by Merrick et al. (2016). The goal state is not the actual state

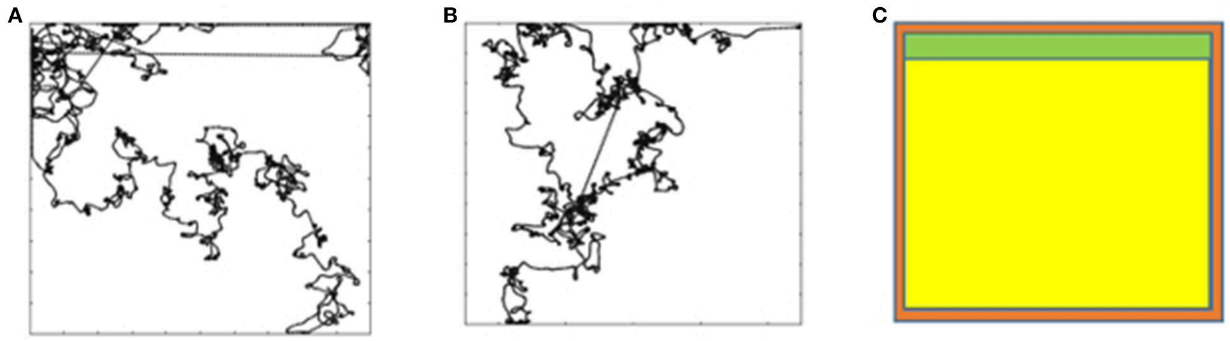


FIGURE 4 | (A) Trajectory for G5. **(B)** Simulation for a G8. **(C)** Likelihood of the reward for G5 and G8.

Table 2 | Experiments and results for approach goals.

ID	M_5	M_6
G^1	32.49% ± 0.64	7.56% ± 0.16
G^2	34.66% ± 0.62	8.00% ± 0.21
G^3	36.58% ± 0.41	8.39% ± 0.14
G^4	35.88% ± 0.43	8.52% ± 0.11
G^5	37.27% ± 0.88	8.84% ± 0.34
G^6	37.25% ± 0.38	8.74% ± 0.19
G^7	36.77% ± 0.57	8.76% ± 0.22
G^8	37.15% ± 0.64	8.73% ± 0.22
G^9	36.71% ± 0.98	8.60% ± 0.26
G^{10}	36.12% ± 0.60	8.24% ± 0.23
G^{11}	36.89% ± 0.86	8.74% ± 0.26
G^{12}	33.58% ± 0.58	7.40% ± 0.17

Table 3 | Experiments and results for avoidance goals.

ID	M_5	M_6	M_7
G^1	36.67% ± 0.32	8.63% ± 0.14	45
G^2	34.88% ± 0.67	8.05% ± 0.25	14
G^3	32.61% ± 0.41	7.53% ± 0.16	12
G^4	33.16% ± 0.53	7.62% ± 0.14	12
G^5	35.60% ± 1.01	8.21% ± 0.30	1
G^6	34.22% ± 0.94	7.95% ± 0.25	16
G^7	33.46% ± 0.55	7.75% ± 0.22	13
G^8	34.90% ± 0.84	8.11% ± 0.18	0
G^9	35.54% ± 0.64	8.31% ± 0.17	0
G^{10}	32.74% ± 0.75	7.52% ± 0.16	6
G^{11}	35.46% ± 0.97	8.26% ± 0.33	0
G^{12}	37.00% ± 0.77	8.56% ± 0.20	7

experienced by the mobile robot but is an event as described by $e_t^i = s_t^i - s_{t-1}^i$. Thus, for an achievement goal type, the aim of the mobile robot is to learn to achieve the transition described by that event, for example, to learn to achieve goal aG^5 listed in **Table 4**, which is to increase speed of both wheels, the robot must learn to increase its right wheel speed by 0.9 and left wheel speed by 0.6 in a single transition of state. The goal is considered achieved when the transition e_t^i is reached regardless of what the state $s_{t-1}^i$is.

Table 4 also shows the results of the experiments (with a 95% confidence interval) for the achievement goals. The e-puck mobile robot simulation was run for 10 trials for each goal with 25,000 steps in each trial. Parameters of Equation (7) were same as in the above experiments, i.e., ρ was set to 0.9, σ set to 1, φ set to −1 and d was the Euclidian distance. Also, the RL exploration parameter epsilon was set to 0.15 with a linear decay schedule. For achievement goals too, when a trial was finished the next trial started at the same position and orientation of the e-puck mobile robot at which the previous trial ended. The Q table, however, was reset after each trial thus there was no learning carried forward between the trials.

While the robot easily achieved goals aG^1 and aG^5, it could either achieve other valid goals only a few times or not able to achieve them at all. Goals aG^2, aG^8, aG^{10}, aG^{11}, aG^{13}, and aG^{17} could be achieved only a few times whereas goals aG^4, aG^9, aG^{14}, aG^{16}, and aG^{21} could not be achieved at all. The reason for that is due to the lack of opportunity. For example, the mobile robot must be near a wall for the event of detecting an obstacle at the front or turning right to avoid an obstacle behind. The argument made in section Maintenance Goal Learning Results regarding reducing the size of the arena to increase the opportunity for learning is valid here too.

Goals aG^3, aG^6, aG^7, and aG^{15} could not be achieved due to the granularity of discretization. For the experiments in this paper, the wheel speed and orientation are discretized into nine values ranging from −π to π. The wheel speed difference for the events for those goals was too small hence when discretized; the values returned are 0 resulting in no change to the wheel speed, i.e., the event of the robot turning left, or right is not detected. For example, consider aG^7 where the goal is to turn right by increasing the right wheel speed by 0.1 (also achieving the change in orientation of −0.1). Discretization of the range of 2π radians into 9 buckets gives the granularity of 0.7 radians, thus making the change of 0.1 radians difficult to detect. This, however, does not mean that the goal is invalid. It is a valid goal, just that, for the robot to be able to learn a goal of such precise transition would require experiments to be run with lower granularity values of wheel speed and orientation, which in turn increases the state space and the size of the Q table and drastically increases the time to learn to achieve those goals.

Figure 5A shows the trajectory for aG^5 (increase speed of both wheels) for one of the trials. The robot learns to attain this goal. In effect, this goal means that the robot has to keep increasing the speed of its wheels. Attaining the maximum speed for both wheels results in the robot not able to achieve the goal anymore and thus receives a negative reward. The robot, however, is again able to attain the goal. This continues until the end of the trial.

Figure 5B shows the trajectory for aG^{22} (turn left) for 25,000 steps. The robot is not able to learn to achieve that goal. The trajectory, however, is surprising, showing long stretches of straight line. We let that trial continue for 100,000 steps, the trajectory for which is shown in **Figure 5C**. The robot still does not learn to achieve the goal. This is because the change in the wheel speed, due to the event (2.0 radians per second for the left wheel speed), is too large for one-time step. In a single step, the maximum change can only be π/2 radians as per the design of the action set. Hence, the goal appears to be unreasonable. The goals aG^{19} and aG^{20} too appear to be unreasonable for the same reason, and as can be seen from **Table 4**, they too could not be achieved. aG^{12} is unreasonable because goal attributes are showing transition for Right and Front-Left sensors without any transition for Front-Right. It is hard to imagine the location of the mobile robot in the arena that will result in such an event. aG^{18} too appears unreasonable because considering the change to the wheel speeds (1.2 and 0.5 radians per second), the transition in the orientation (−0.1 radians) is too small.

Either such unreasonable events experienced by the robot during the experience gathering stage in the experiments run by Merrick et al. (2016) could be due to noise, delay in sensing or

Table 4 | Experiments and results for achievement goals.

ID	Goal attributes	Meaning of goal	M_2	Is goal valid?
aG^1	(0.0, 0.0, 0.0, 0, 0, 0, 0, 0, 0)	Achieve no change	25000 ± 0	Yes
aG^2	(0.0, 0.0, 0.0, 0, 0, 1, 0, 0, 0)	Detect obstacle in front	43 ± 21	Yes
aG^3	(−0.1, 0.0, 0.0, 0, 0, −1, 0, 0, 0)	Turn left to avoid obstacle on the right	0 ± 0	Yes
aG^4	(−0.6, 0.0, −0.1, 0, 0, 0, −1, 0, 0)	Turn left to avoid obstacle on the right	0 ± 0	Yes
aG^5	(0.9, 0.6, 0.0, 0, 0, 0, 0, 0, 0)	Increase speed of both wheels	6521 ± 268	Yes
aG^6	(−0.1, 0.1, 0.1, 0, 0, 0, 0, 0, 0)	Turn left	0 ± 0	Yes
aG^7	(0.1, 0.0, −0.1, 0, 0, 0, 0, 0, 0)	Turn right	0 ± 0	Yes
aG^8	(0.1, −0.4, 0.0, 0, 0, 0, 0, −1, −1)	Turn right to avoid obstacle behind	54 ± 17	Yes
aG^9	(−0.3, 0.4, −0.3, 0, 0, −1, −1, 0, 0)	Turn left to avoid obstacle on the right	0 ± 0	Yes
aG^{10}	(0.0, 0.5, 0.2, 0, 0, 1, 0, 0, 0)	Turn left to detect obstacle on the right	29 ± 16	Yes
aG^{11}	(−0.6, −0.8, −0.2, 0, 0, −1, 0, 0, 0)	Turn right to avoid obstacle	10 ± 4	Yes
aG^{12}	(0.0, 0.7, 0.3, 0, −1, 1, 0, 0, 0)	Turn left to sense obstacle on right	0 ± 0	No
aG^{13}	(0.2, −0.8, −0.4, 0, 0, 0, 0, 1, 0)	Turn right to sense obstacle on left	12 ± 4	Yes
aG^{14}	(0.0, 0.6, 0.1, 0, 0, 0, 0, 1, 1)	Turn to detect obstacle behind	0 ± 0	Yes
aG^{15}	(0.0, −0.1, 0.0, 0, 1, 1, 0, 0, 0)	Turn right to sense obstacle in front	0 ± 0	Yes
aG^{16}	(1.0, 0.5, 0.1, 0, 1, 0, 0, 0, 0)	Turn right to sense obstacle on left	0 ± 0	Yes
aG^{17}	(0.7, 0.9, 0.3, 0.0, −1, 0, 0, 0, 0)	Turn left to sense obstacle on left	18 ± 3	Yes
aG^{18}	(1.2, 0.5, −0.1, 0, −1, 0, 0, 0, 0)	Turn to avoid obstacle on left	0 ± 0	No
aG^{19}	(0.2, 2.7, −0.2, 0, −1, 0, 0, 0, 0)	Turn to avoid obstacle on left	0 ± 0	No
aG^{20}	(−1.7, −0.5, 0.1, 0, 1, 0, 0, 0, 0)	Turn to detect obstacle on right	0 ± 0	No
aG^{21}	(−0.7, −1.2, −0.3, 0, 1, 0, 0, 0, 0)	Turn to detect obstacle on left	0 ± 0	Yes
aG^{22}	(1.4, 2.0, 0.2, 0, 0, 0, 0, 0, 0)	Turn left	0 ± 0	No

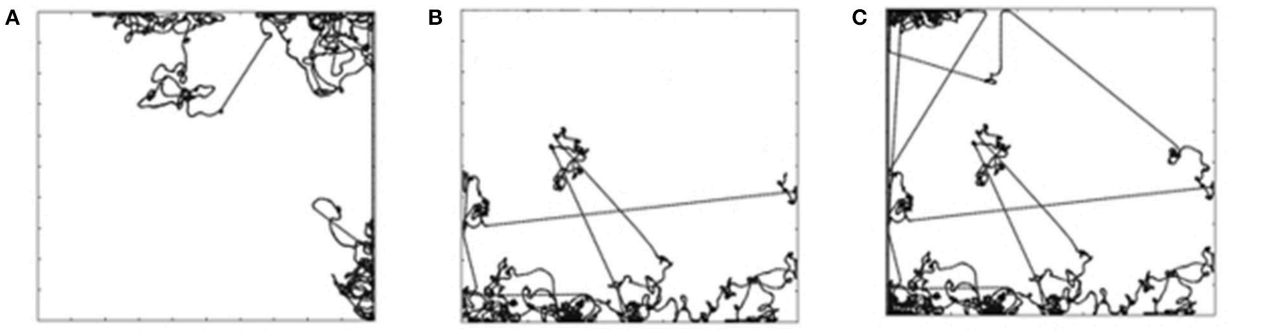


FIGURE 5 | (A) Trajectory for aG^5. **(B)** Trajectory for aG^{22} (run for 25,000 steps). **(C)** Simultation for a aG^{22} run for 100,000 steps.

that the mobile robot might have got stuck and then unstuck to the wall resulting in an invalid event ($e_t = s_t - s_{t-1}$) or that the unreasonable events were due to an error in clustering, resulting in cluster centroid not being a correct representative of the cluster. If latter was the case, then it requires reanalysis of the generated clusters. Possible solutions to rectify the incorrect representation of the cluster centroid could be to place a minimum threshold on the cluster size or to shift the cluster centroids to the nearest valid attribute value. In any case, those goals appear unreasonable and are marked as invalid in the table. Based on the findings of the above experiments, for the experiments in the next section, we have removed the orientation attribute from the RL state vector, reduced the size of the arenas and, not used any of the invalid goals.

DEMONSTRATION OF HOW PRIMITIVE GOAL-BASED REWARD FUNCTIONS CAN BE COMBINED

Not all tasks can be represented as a single goal type. Consider an example detailed in Dastani and Winikoff (2011), if the task for a personal assistant agent that manages a user's calendar is to book a meeting, it can be represented as an achievement goal, however people's schedules change and hence to ensure that the meeting invite remains in the calendar of all the participants, the task is better modeled by a combination of goal types. The goal can be represented as "achieve then maintain" where the aim is to achieve the goal and then maintain it. As another example, consider a wall following mobile robot. The robot has to first approach a wall and then maintain a set distance from the wall either to its left or to its right side. This goal can be represented as "approach then maintain" where the aim of the mobile robot is to first approach the goal state (i.e., a wall to its left or right) and then maintain it. We term this as a compound goal-based reward function, as it can be built from multiple primitive goal-based reward functions.

In this section, we demonstrate compound goal-based reward functions constructed using if-then rules to trigger different primitive reward functions in different states. In this paper, the if-then rules are hand-crafted as we aim to demonstrate that primitive reward functions can be combined to motivate learning of complex behaviors. The question of how to do this autonomously is discussed as an avenue for future work in Section Autonomous Generation of Compound Reward Functions and Conditions for Goal Accomplishment.

Experimental Setup

To demonstrate compound goal-based reward functions, we use the e-puck robot in three new environments. The environments are as shown in **Figures 6A–C**. The maze environment, shown in **Figure 6A**, has walls to form a simple maze. In this environment, the goal of the robot is to follow a wall. That goal is actually a compound goal. In order to achieve the goal, the robot has to learn primitive goals detailed in **Tables 1–4**. The compound **Function 1** details the if-then rules to achieve this goal. The environment with obstacles, shown in **Figure 6B**, has cylindrical and cuboid objects that act as obstacles. The goal of the robot is to learn to avoid obstacles. The compound **Function 2** details the if-then rules to achieve that goal. The third environment is shown in **Figure 6C** is a circular arena with tracks. The goal of the robot is to learn to follow a track which is detailed by compound **Function 3**. Experiments were run for the following goals expressed using compound goal-based reward functions. The primitive reward functions shown in the if-then rules (**Function 1**, **Function 2** and **Function 3**) are the same as in **Tables 1–4**.

We use the same Dyna-Q algorithm that is detailed in section Experiments for Maintenance, Approach, Avoidance, and Achievement Goal Types. Action selection was using the epsilon-greedy method with epsilon parameter set to 0.1 throughout the learning process. 10 trials were run for each of the experiment with each trial consisting of 25,000 steps.

The state space for this robot is different from that in section Experiments for Maintenance, Approach, Avoidance, and Achievement Goal Types. In addition to the six distance sensors as detailed in the experiments in section Experiments for Maintenance, Approach, Avoidance, and Achievement Goal Types, we also use the ground sensors for these experiments. We label the three ground sensors as *Ground-Left, Ground-Centre, Ground-Right*. The state of the mobile robot comprises of following parameters: left wheel direction, right wheel direction, left sensor value, right sensor value, front-left sensor value, front-right sensor value, rear-left sensor value, rear right sensor value, ground left sensor value, ground center sensor value and ground right sensor value. The state is a vector represented by $[\omega^R\ \omega^L\ s^L\ s^R\ s^{FL}\ s^{FR}\ s^{RL}\ s^{RR}\ s^{GL}\ s^{GC}\ s^{GR}]$. ω^R and ω^L are the rotational velocities of the right and the left wheels that are discretized to binary values with 1 indicating that the wheel is moving forward and 0 indicating that it is moving backwards. For the proximity sensors, we use binary values with 0 indicating that there is no object in the proximity of the sensor and 1 indicates that the object is near. For ground sensors as well, we use binary values with 0 indicating that the sensor is

Function 1 | Wall following goal in the maze arena.

```
if obstacle on the left
  $aG^{17}$ – achieve turning left
elseif obstacle close on the left
  $G^1$ – maintain moving forward
elseif obstacle on the right
  $aG^{11}$ - achieve turning right
elseif obstacle close on the right
  $G^1$ – maintain moving forward
elseif obstacle at the front and left /*i.e,corner on the left */
  achieve turning right
elseif obstacle at the front and right /*i.e,corner on the right */
  achieve turning left
elseif obstacle at the front
  $aG^{11}$ - achieve turning right
elseif no obstacle nearby
  $G^1$ – maintain moving forward
end
```

Function 2 | Obstacle avoidance goal in the arena with obstacles.

```
if obstacle on the left
  $aG^{13}$ – achieve turning right
elseif obstacle on the right
  $aG^4$ - achieve turning left
elseif obstacle at the front and/or side
  $aG^{11}$ - achieve turning right
elseif obstacle at the back
  $G^1$ – maintain moving forward
elseif no obstacle anywhere nearby
  $G^1$ – maintain moving forward
end
```

Function 3 | Track following goal in the circular arena with tracks.

```
if the obstacle anywhere nearby
  $aG^{11}$ - achieve turning right
elseif track to the left
  achieve turning left
elseif track to the right
  achieve turning right
elseif on the track
  $G^1$ – maintain moving forward
end
```

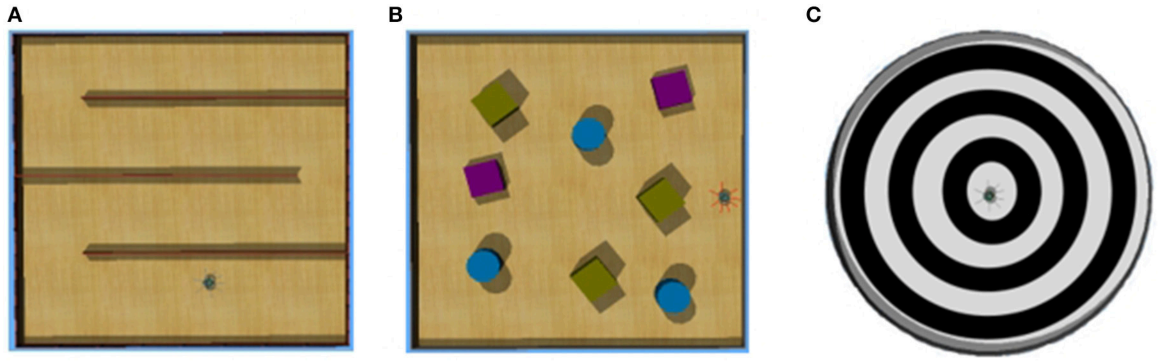


FIGURE 6 | (A) Maze arena. **(B)** Arena with obstacles. **(C)** A circular arena with tracks.

Table 5 | Results for compound goals.

ID	Goal Description	M_1	M_2	M_3	M_4
G^1	Wall following	1373 ± 29	16833 ± 115	10 ± 0	78 ± 6
G^2	Avoiding obstacles	747 ± 24	13613 ± 109	11 ± 0	81 ± 8
G^3	Following a track	992 ± 24	14634 ± 127	9 ± 0	74 ± 8

detecting light color and 1 indicating that it is indicating dark color.

The action space comprises of three values: 1–turn left, 2–move forward, and 3–turn right.

Results

Table 5 shows the results of the wall following, obstacle avoidance, and track following goals. Results were averaged over 10 trials, and its standard deviation is shown. The metrics used to measure agent's performance are the same as the ones defined in section Primitive Goal-based Motivated Reward Functions however here the metrics M_1, M_2, M_3 and M_4 measure cumulative reward gained by the agent for all the primitive goals combined, i.e., the measurement for the compound goal-based reward.

Figure 7 shows the trajectory for one of the trials of the mobile robot learning to follow the wall using compound goal-based reward function (**Function 1**). The function comprises of a combination of achievement and maintenance goal types each of which are triggered in a specific situation. When there is no wall in the proximity, the robot is learning to move forward. Once it is near the wall (either to the left or the right), it learns to follow the wall on that side. When it reaches the edge of the wall, it is not able to follow it around for the initial two or three attempts however eventually learns to follow the wall around and continues to follow the wall as shown in the zoomed-in section of **Figure 7**. Trajectory labeled 4 in the zoomed-in section of **Figure 7** is the one where the agent follows the wall all the way around.

Figure 8A shows the trajectory for one of the trials of the mobile robot learning to avoid obstacles using the compound goal-based reward function (**Function 2**). This function too comprises a combination of achievement and maintenance goal types each of which are triggered in a specific situation. When there is no obstacle nearby, the robot has to learn to move forward. When it is close to an obstacle, it has to learn to turn right and when it has the obstacle at its back it has to learn to move forward, thus moving away from the obstacle. **Figure 8B** shows the trajectory for one of the trials of the mobile robot learning to follow a track using the compound goal-based reward function (**Function 3**). When the robot has a wall in its proximity, it has to learn to turn right. When near the track, it has to learn to turn toward the track such that it is entirely on the track. Once on the track, it has to learn to move forward.

CONCLUSION AND FUTURE WORK

This paper proposed reward functions for reinforcement learning based on the type of goal as categorized by the Belief Desire Intension community. The reward functions for the maintenance, approach, avoidance, and achievement goal types exploit the inherent property of its type, making them task-independent. Using simulated e-puck mobile robot experiments, we show how these intrinsic reward functions bridge the gap between autonomous goal generation and goal learning thus endowing the robot with the capability to learn in an autonomous and open-ended manner.

We present metrics to measure the agent's performance. The measurements show that using the proposed reward functions; all the valid goals will be learnt, some slower than the others due to the lack of opportunity. The goals that are not learnt are either very difficult to learn, unreasonable or invalid. The results also highlight the importance of attributes used in the design of the state vector as it can severely limit the learning opportunity, for example, usage of orientation attribute in the state vector. Although, this paper does not make any claim whether for or against any goal generation techniques, in the future work, the findings from this paper could be used to tune the goal generation

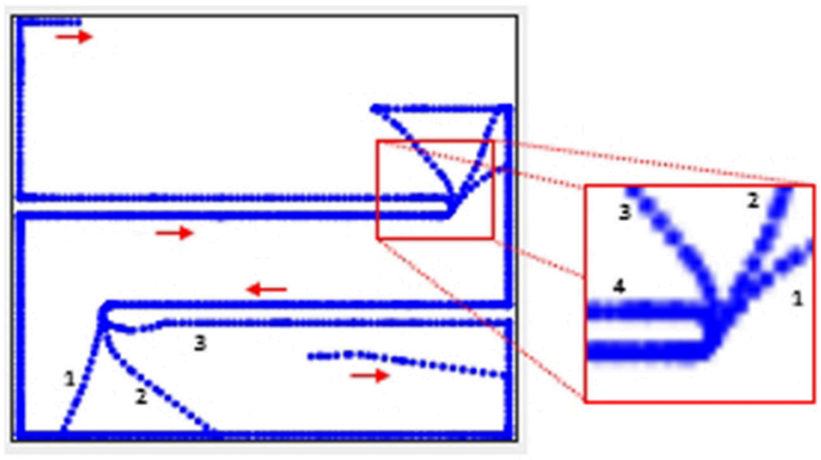


FIGURE 7 | Trajectory for wall following goal in the maze arena.

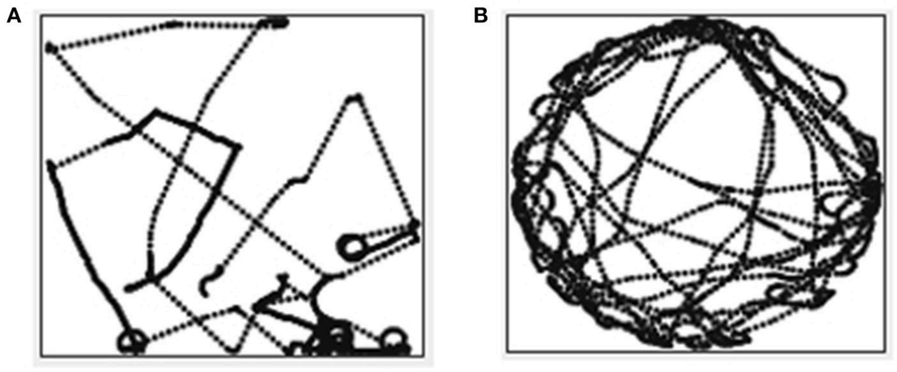


FIGURE 8 | (A) Trajectory for obstacle avoidance goal in the arena with obstacles. **(B)** Trajectory for track following goal in the arena with tracks.

technique used by Merrick et al. (2016). We also show that the maintenance goals are easier to learn than the achievement goals. Approach and avoidance goals are even easier due to their inherent nature. This is because, for the maintenance goal, the agent is rewarded only when it can maintain the distance below a certain threshold, whereas, for approach and avoidance goals, the agent is rewarded for the approach or the avoidance attempt irrespective of its distance from the goal.

We further show how rather than treating the goal of a single type, the agent can decide whether it wants to maintain, approach, avoid or achieve the goal based on the situation it is experiencing. This situation specific goal type usage means the agent now knows what it has to learn in a specific situation thus directing the learning. A compound goal-based reward function can be designed by chaining any number of primitive reward functions. This raises following directions for future work.

Autonomous Generation of Compound Reward Functions

This paper demonstrated that primitive goal-based reward functions could be combined using if-then rules to create learnable compound reward functions. However, this raises a question whether it is possible for an agent to self-generate such rules or some other means of combining the primitive reward functions. One potential solution could be for the agent to autonomously determine the structure or regions in its state space each of which relates to a primitive goal. Merrick et al. (2016) have shown how the history of experienced states can be used to generate the goals. In a similar fashion, a coarse level clustering can be run on the experienced states to form these regions in the state space. Once those regions are known, one can then map the regions (primitive goal) with the goal state (compound goal) to enable the generation of the if-then rules. A formal framework is required for identifying complementary or conflicting goals so that complementary goals can be formed into compound reward functions and conflicting goals avoided.

Conditions for Goal Accomplishment

We also saw in this work that the agents learn solutions to some goals more effectively when they are in certain situations where the conditions support learning of that particular goal. This suggests that there is a role for concepts such as opportunistic learning (Graham et al., 2012) to maximize the efficiency of learning such that the agent only attempts goals that are feasible in a given situation.

AUTHOR CONTRIBUTIONS

PD and KM conceived of the presented concept and planned the experiments. PD carried out the experiments under the supervision of KM and IR. PD wrote the manuscript in consultation with KM, IR, and NS. All authors discussed the results, provided critical feedback and contributed to the final version of the manuscript.

REFERENCES

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete Problems in AI Safety. *arXiv [preprint]*. arXiv:1606. 06565.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., et al. (2017). "Hindsight Experience Replay," in *Advances in Neural Information Processing Systems,* (Long Beach, CA), 5048–5058.

Baldassarre, G., and Mirolli, M. (eds.). (2013). *Intrinsically Motivated Learning in Natural and Artificial Systems*. Heidelberg: Springer.

Baraldi, A. (1998). *Simplified ART: A New Class of ART Algorithms*. Berkeley, CA: International Computer Science Institute.

Baranes, A., and Oudeyer, P.-Y. (2010a). "Intrinsically motivated goal exploration for active motor learning in robots: A case study," in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings* (Taipei), 1766–1773.

Baranes, A., and Oudeyer, P.-Y. (2010b). "Maturationally-constrained competence-based intrinsically motivated learning," in *2010 IEEE 9th International Conference on Development and Learning* (Ann Arbor, MI), 197–203.

Barto, A. G., Mirolli, M., and Baldassarre, G. (2013). Novelty or Surprise? *Front. Psychol.* 4:907. doi: 10.3389/fpsyg.2013.00907

Bonarini, A., Lazaric, A., and Restelli, M. (2006). "Incremental skill acquisition for self-motivated learning animats,"in *Proceedings of the Ninth International Conference on Simulation of Adaptive Behavior (SAB-06)* 4095 (Rome), 357–368.

Braubach, L., Pokahr, A., Moldt, D., and Lamersdorf, W. (2005). "Goal representation for BDI agent systems," in *Second International Workshop on Programming Multiagent Systems: Languages and Tools* (Utrecht), 9–20. doi: 10. 1007/978-3-540-32260-3_3

Dastani, M., and Winikoff, M. (2011). "Rich goal types in agent programming," in *In The 10th International Conference on Autonomous Agents and Multiagent Systems* (Taipei), 405–412.

Dewey, D. (2014). "Reinforcement learning and the reward engineering principle,"in *AAAI Spring Symposium Series* (Stanford, CA), 1–8.

Duff, S., Harland, J., and Thangarajah, J. (2006). "On proactivity and maintenance goals,"in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems - AAMAS'*06:1033 (Hakodate).

Elliot, A. J. (2008). *Handbook of Approach and Avoidance Motivation*. New York, NY: Psychology Press. doi: 10.1017/CBO9781107415324.004

Graham, J. T., Starzyk, J. A., and Jachyra, D. (2012). "Opportunistic motivated learning agents," in *International Conference on Artificial Intelligence and Soft Computing* (Napoli).

Held, D., Geng, X., Florensa, C., and Abbeel, P. (2017). *Automatic Goal Generation for Reinforcement Learning Agents*. Available online at: http://arxiv.org/abs/ 1705.06366

Hindriks, K. V., and Van Riemsdijk, M. B. (2007). "Satisfying maintenance goals," in *International Workshop on Declarative Agent Languages and Technologies* (Honolulu, HI), 86–103.

Laud, A., and DeJong, G. (2002). "Reinforcement learning and shaping: encouraging intended behaviors," in *Proceedings of International Conference on Machine Learning* (Sydney, NSW), 355–362.

Merrick, K. E. (2007). *Modelling Motivation For Experience-Based Attention Focus In Reinforcement Learning*. School of Information Technologies, University of Sydney.

Merrick, K. E. (2012). "Intrinsic motivation and introspection in reinforcement learning,"in *IEEE Transactions on Autonomous Mental Development* 4, 315–329.

Merrick, K. E., and Maher, M. L. (2009). "Motivated reinforcement learning: curious characters for multiuser games," in *Motivated Reinforcement Learning: Curious Characters for Multiuser Games* (Berlin, Heidelberg: Springer), 1–206.

Merrick, K. E., Siddique, N., and Rano, I. (2016). Experience-based generation of maintenance and achievement goals on a mobile robot. *Paladyn J. Behav. Robot.* 7, 67–84. doi: 10.1515/pjbr-2016-0006

Mirolli, M., and Baldassarre, G. (2013). "Functions and mechanisms of intrinsic motivations. the knowledge versus competence distinction," in *Intrinsically Motivated Learning in Natural and Artificial Systems*, 49–72.

Nehmzow, U., Gatsoulis, Y., Kerr, E., Condell, J., Siddique, N., and McGuinnity, T. M. (2013). "Novelty detection as an intrinsic motivation for cumulative learning robots," in *Intrinsically Motivated Learning in Natural and Artificial Systems,* (Berlin; Heidelberg: Springer), 185–207.

Neto, H. V., and Nehmzow, U. (2004). "Visual novelty detection for inspection tasks using mobile robots," in *Towards Autonomous Robotic Systems: Proceedings of the 5th British Conference on Mobile Robotics (TAROS'04)* (University of Essex).

Ng, A. Y., Harada, D., and Russell, S. (1999). "Policy invariance under reward transformations : theory and application to reward shaping," in *Sixteenth International Conference on Machine Learning* 3, (Bled), 278–287.

Oudeyer, P.-Y., and Kaplan, F. (2007). What is intrinsic motivation? A typology of computational approaches. *Front. Neurorobot.* 1:6. doi: 10.3389/neuro.12.006. 2007

Oudeyer, P.-Y., Kaplan, F., and Hafner, V., V (2007). "Intrinsic motivation systems for autonomous mental development,"in *IEEE Transactions On Evolutionary Computation* 2, 265–286.

Rao, A. S., and Georgeff, M. P. (1995). BDI agents: from theory to practice. *ICMAS* 95, 312–319.

Regev, G., and Wegmann, A. (2005). "Where do goals come from : the underlying principles of goal-oriented requirements engineering," in *International Conference on Requirements Engineering* (Paris), 353–362.

Rolf, M., Steil, J. J., and Gienger, M. (2010). "Bootstrapping inverse kinematics with goal babbling,"in *2010 IEEE 9th International Conference on Development and Learning, ICDL-2010 - Conference Program* (Ann Arbor, MI), 147–154.

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2010). "Biological cumulative learning through intrinsic motivations: a simulated robotic study on the development of visually-guided reaching,"in *Proceedings of the Tenth International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems* (Lund), 121–128.

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2012). "Intrinsic motivation mechanisms for competence acquisition," in *IEEE International Conference on Development and Learning* (San Diego, CA), 1–6.

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2016). GRAIL: a goal-discovering robotic architecture for intrinsically-motivated learning. *IEEE Trans. Cogn. Dev. Syst.* 8, 214–231. doi: 10.1109/TCDS.2016.2538961

Singh, S., Barto, A. G., and Chentanez, N. (2005). "Intrinsically motivated reinforcement learning," in *18th Annual Conference on Neural Information Processing Systems (NIPS)* (Vancouver, BC), 1281–1288.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge: MIT Press.

Thrun, S. B., and Mitchell, T. M. (1995). Lifelong robot learning. *Robot. Autonom. Syst.* 15, 25–46. doi: 10.1016/0921-8890(95)00004-Y

Van Lamsweerde, A. (2001). "Goal-oriented requirements engineering: a guided tour," in *Proceedings Fifth IEEE International Symposium on Requirements Engineering,* (Toronto), 249–262.

Van Riemsdijk, M. B., Dastani, M., and Winikoff, M. (2008). "Goals in agent systems: a unifying framework," in *Proceedings of 7th International Conference on Autonomous Agents and Multiagent Systems* (Estoril), 713–720.

Weng, J., Mcclelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., et al. (2001). Autonomous mental development by robots and animals. *Science* 291, 599–600. doi: 10.1126/science.291.5504.599

Conflict of Interest Statement: PD, KM, IR, and NS declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 09 October 2018
doi: 10.3389/fnbot.2018.00065

Evolving Robust Policy Coverage Sets in Multi-Objective Markov Decision Processes Through Intrinsically Motivated Self-Play

Sherif Abdelfattah*, Kathryn Kasmarik and Jiankun Hu

School of Engineering and Information Technology, University of New South Wales, Canberra, ACT, Australia

Many real-world decision-making problems involve multiple conflicting objectives that can not be optimized simultaneously without a compromise. Such problems are known as multi-objective Markov decision processes and they constitute a significant challenge for conventional single-objective reinforcement learning methods, especially when an optimal compromise cannot be determined beforehand. Multi-objective reinforcement learning methods address this challenge by finding an optimal coverage set of non-dominated policies that can satisfy any user's preference in solving the problem. However, this is achieved with costs of computational complexity, time consumption, and lack of adaptability to non-stationary environment dynamics. In order to address these limitations, there is a need for adaptive methods that can solve the problem in an online and robust manner. In this paper, we propose a novel developmental method that utilizes the adversarial self-play between an intrinsically motivated preference exploration component, and a policy coverage set optimization component that robustly evolves a convex coverage set of policies to solve the problem using preferences proposed by the former component. We show experimentally the effectiveness of the proposed method in comparison to state-of-the-art multi-objective reinforcement learning methods in stationary and non-stationary environments.

Keywords: multi-objective optimization, intrinsic motivation, adversarial, self-play, reinforcement learning, Markov process, decision making

OPEN ACCESS

Edited by:
Pierre-Yves Oudeyer, Institut National de Recherche en Informatique et en Automatique (INRIA), France

Reviewed by:
Antoine Cully, Imperial College London, United Kingdom
Eiji Uchibe, Advanced Telecommunications Research Institute International (ATR), Japan

****Correspondence:***
Sherif Abdelfattah
sherif.abdelfattah@student.adfa.edu.au

Received: *30 May 2018*
Accepted: *18 September 2018*
Published: *09 October 2018*

Citation:
Abdelfattah S, Kasmarik K and Hu J (2018) Evolving Robust Policy Coverage Sets in Multi-Objective Markov Decision Processes Through Intrinsically Motivated Self-Play. Front. Neurorobot. 12:65. doi: 10.3389/fnbot.2018.00065

1. INTRODUCTION

Reinforcement learning (RL) is a learning paradigm that works by interacting with the environment in order to evolve an optimal policy (action selection strategy) guided by the objective to maximize the return of a reward signal (Sutton and Barto, 1998). Recently, deep reinforcement learning (DRL) benefit from the automatic hierarchical features extraction and complex functional approximation of deep neural networks (DNNs) (LeCun et al., 2015). This has led to many breakthroughs (Mnih et al., 2015; Silver et al., 2016, 2017) in solving sequential decision-making problems fulfilling the Markov property [known as Markov decision processes (MDPs)]. While the majority of problems addressed by DRL methods involve only one objective of maximizing a scoring function (e.g., score in an Atari game, or the game of Go), many real-world problems constitute multiple conflicting objectives that cannot be optimized simultaneously without a tradeoff (prioritization) among the defined objectives. Take a search and rescue task as an example in which a robot has to maximize the number of victims found, minimize exposure to fire risk to avoid destruction, and minimize the total task time. Another example could be a patrolling drone aiming at maximizing the area of the scanned region, maximizing the number of detected objects of interest, and maximizing battery life. Such problems are known as multi-objective Markov decision processes (MOMDPs)[1].

Multi-objective reinforcement learning (MORL) extends the conventional RL paradigm to accept multiple reward signals instead of a single reward signal, each one is dedicated to an objective (Roijers and Whiteson, 2017). Basically, MORL methods fall into two broad groups: single policy group, and multiple policy group (Roijers and Whiteson, 2017). In the former group, it is assumed that the user's preference is defined before solving the problem, therefore, it can be used to transform it into a single objective problem using scalarization functions. Albeit, this assumption can be difficult to satisfy in many practical scenarios. Alternatively, the latter group aims at finding a set of optimal policies that can satisfy any user's preference in solving the problem. In order to achieve this, these methods perform an intensive search process using an environment's model to find such a set of policies. This makes them difficult to operate in an online manner and to efficiently adapt to non-stationary dynamics in the environment.

In this paper, we do not assume the existence of an optimal user's preference beforehand, so we will consider the multiple policy MORL approach. In order to deal with the limitations of this approach, we look at the two building blocks of these methods depicted in **Figure 1**: the preference exploration component; and the policy coverage set optimization component. Currently, preference exploration is achieved through random exploration in evolutionary methods (Busa-Fekete et al., 2014), or by systematic heuristic approaches such as the optimistic linear support (OLS) (Roijers et al., 2014). However, these approaches adopt exhaustive search scheme that can not adapt efficiently to the non-stationary dynamics in the environment. In order to overcome these limitations, our proposed intrinsically motivated preference exploration component targets three main characteristics. First, it actively explores preferences that contribute to the large mass of uncertainty about the policy coverage set's performance. Second, it performs this exploration automatically guided by an intrinsic reward signal. Third, it can adapt to non-stationary dynamics in the environment by revisiting the affected preference areas. While for the policy coverage set optimization component, we utilize the concept of policy bootstrapping using steppingstone policies. Basically, this concept is based on the assumption that while there is a large number of policies each is specialized for a specific preference, there is a smaller number of steppingstone policies that can bootstrap policies within intervals of preferences. By targeting steppingstone policies instead of specialized policies during the evolution of the policy coverage set, we can adapt robustly to non-stationary dynamics in the environment.

[1]This should not be confused with mixed observability Markov decision processes abbreviated similarly as MOMDPs.

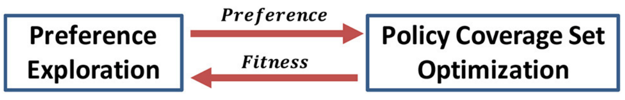


FIGURE 1 | A block diagram for a multiple policy MORL approach for solving MOMDPs.

In this paper, we address the MOMDP problem through an adversarial intrinsically motivated self-play approach. Our contribution comes into three folds. First, we propose a novel preference exploration technique based on knowledge-seeking intrinsic motivation. Second, we propose a novel algorithm for fuzzy policy bootstrapping to developmentally evolve the policy coverage set in MOMDP problems. Third, we experimentally evaluate the performance of our proposed method using common multi-objective environments in MORL literature and comparing to the state-of-the-art MORL methods.

The rest of this paper is organized as follows. Section 2 introduces the background concepts. Section 3 reviews the related literature. Section 4 describes our proposed method. Section 5 illustrates our experimental design. Section 6 presents the results and discusses the findings. Finally, section 7 concludes the work and indicates the future work.

2. BACKGROUND

In this section, we are going to introduce the related background concepts and the research problem definition.

2.1. Multi-Objective Optimization

In a multi-objective optimization problem there are multiple objectives that are naturally in conflict with each other and can not be optimized simultaneously without a compromise (Deb, 2014). The problem can be mathematically formulated as follows:

$$\begin{aligned} &\max\left(R^{1}\left(\pi\right), R^{2}\left(\pi\right), \ldots, R^{M}\left(\pi\right)\right) \\ &s.t.\, g^{j}\left(\pi\right) \leq 0,\ j = 1, 2, \ldots, J \end{aligned} \qquad (1)$$

The aim is to optimize (maximize or minimize) a set of reward functions $\{R^1(\pi), R^2(\pi), \ldots, R^M(\pi)\}$, where each function is dedicated to a single objective $o^m\,(m = 1, 2, \ldots, M)$, the parameter $\pi \in \Pi$ represents the policy parametrization (decision variables) to be optimized over the parameters search space Π, and the set $\{g^1(\pi), g^2(\pi), \ldots, g^J(\pi)\}$ represents the defined constraint functions of the problem.

In order to find the coverage set of policies that can satisfy any user's preference in solving the problem, a search procedure has to find and rank policies based on the dominance over the defined objectives.

Definition 2.1. Dominance: A solution (A) dominates solution (B) if (A) is better than (B) for at least one objective and is equal to (B) for all other objectives.

For further illustration of Definition 2.1, **Figure 2** shows the solution space for a two-objective problem. It can be noticed

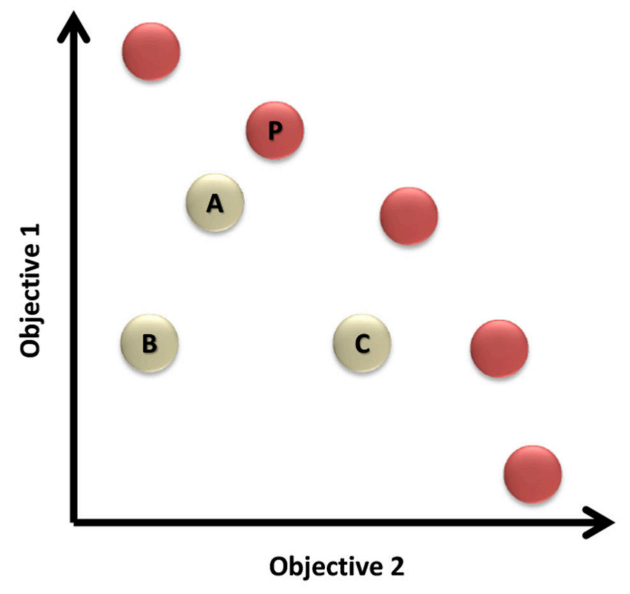


FIGURE 2 | The solution space of a two-objective problem. The red circles are representing the set of non-dominated solutions known as the Pareto front.

that solutions (A) and (C) dominate solution (B), while the set of solutions represented in red circles are the Pareto front of non-dominated solutions in this problem.

Definition 2.2. Pareto Front: The Pareto front is the set of non-dominated solutions that solves the multi-objective problem.

The Pareto front is represented by the red dots in the example illustrated by **Figure 2**.

Definition 2.3. Preference: A preference is defined as a weight vector with each weight element dedicated to a specific objective $\vec{w} = \left[w^1, w^2, \ldots, w^M\right]^T$, such that the sum of all the elements equals one $\sum_{m=1}^{M} w^m = 1$.

Definition 2.4. Scalarization Function: A scalarization function h, transforms a vector of multiple objectives' values into a single objective scalar value given a preference as parameter $o_{\vec{w}} = h(\vec{o}, \vec{w})$.

When the scalarization function is linear or piecewise linear, the front shaped by intersecting functions parametrized by different preferences is the convex hull (CH).

Definition 2.5. Convex Hull: A convex hull is a subset of the policy space (Π) that contains optimal policies that can match any user's preference:

$$CH(\Pi) = \left\{\pi : \pi \in \Pi \wedge \exists \vec{w} \forall \left(\pi' \in \Pi\right) \vec{w} \cdot \vec{r}^{\pi} \geq \vec{w} \cdot \vec{r}^{\pi'}\right\}$$

We illustrate graphically the CH concept for a linear scalarization function over two objectives in **Figure 3**. In **Figure 3A**, the two axes indicate the normalized reward values for each objective. The CH is represented by the convex solid line surface that includes all the red dots. While the Pareto front is represented by the non-convex surface drawn by dashed and solid lines that includes all the red and blue points. The red dots represent undominated policies that fall in the CH. The blue dots represent the undominated policies that fall outside the CH and within the Pareto front. The black dots represent dominated policies. Given a linear scalarization function, **Figure 3B** shows the scalarized reward output (a line) for each different preference. We depict the first weight component (w_1) on the x-axis ($w_2 = 1 - w_1$), and the scalarized reward value on the y-axis. The set of optimal policies that lie in the CH can be found in the surface represented by black bold lines in **Figure 3B**. This upper surface is a piecewise linear and convex function.

The CH surface can contain excess policies (Roijers et al., 2013). Therefore, we can define a subset of it that contains the minimal number of unique policies that solve the problem.

Definition 2.6. Convex Coverage Set: A convex coverage set (CCS) is a subset of the CH that can provide for each preference ($\vec{w}$) a policy whose scalarized reward value is maximal:

$$CCS\left(\Pi\right) \subseteq CH\left(\Pi\right) \wedge$$
$$\left(\forall \vec{w}\right)\left(\exists \pi\right)\left(\pi \in CCS\left(\Pi\right) \wedge \forall \left(\pi' \in \Pi\right) \vec{w} \cdot \vec{r}^{\pi} \geq \vec{w} \cdot \vec{r}^{\pi'}\right)$$

2.2. Multi-Objective Markov Decision Processes

Markov decision processes (MDPs) formulate a sequential decision making framework in which an agent observes the environment's state (s_t) at time t, takes an action (a_t), transits to a new state (s_{t+1}), and gets a reward value (r_{t+1}) for being in the new state (Papadimitriou and Tsitsiklis, 1987). A multi-objective Markov decision process (MOMDP) extends this sequential decision making framework by allowing a vector of reward signals to be passed to the agent after transiting to the new state (Roijers et al., 2013). The difference between a MDP and MOMDP is depicted in **Figure 4**. The MOMDP formalism is represented by a tuple $\left\langle S, A, \mathbb{P}_{ss'}, \vec{R}, \mu, \gamma\right\rangle$, where S is the state space, A is the action space, $\mathbb{P}_{ss'} = \Pr(s_{t+1} = s'|s_t = s, a_t = a)$ is the state transition probability, $\vec{R} \in \mathbb{R}^M \forall R : S \times A \times S' \rightarrow r \in \mathbb{R}$ is the vector of reward functions dedicated to M number of objectives, $\mu = \Pr(s_0)$ is the probability distribution of initial states, and $\gamma \in [0, 1)$ is the discounting factor for the influence of the future rewards.

The objective of the learning agent is to maximize the expected scalarized reward return starting from time t using a scalarization function h given a user's preference $\vec{w}$:

$$R_t^{\vec{w}} = \sum_{l=0}^{T} \gamma^l h\left(\vec{r}_{t+l+1}, \vec{w}\right) \tag{2}$$

where T constitutes the *time horizon* which is equal to ∞ in the *infinite time horizon* scenario.

2.3. Problem Definition

Given a MOMDP problem formalism $\left\langle S, A, \mathbb{P}_{ss'}, \vec{R}, \mu, \gamma\right\rangle$, we need to find the CCS with the minimal cardinality that maximizes the

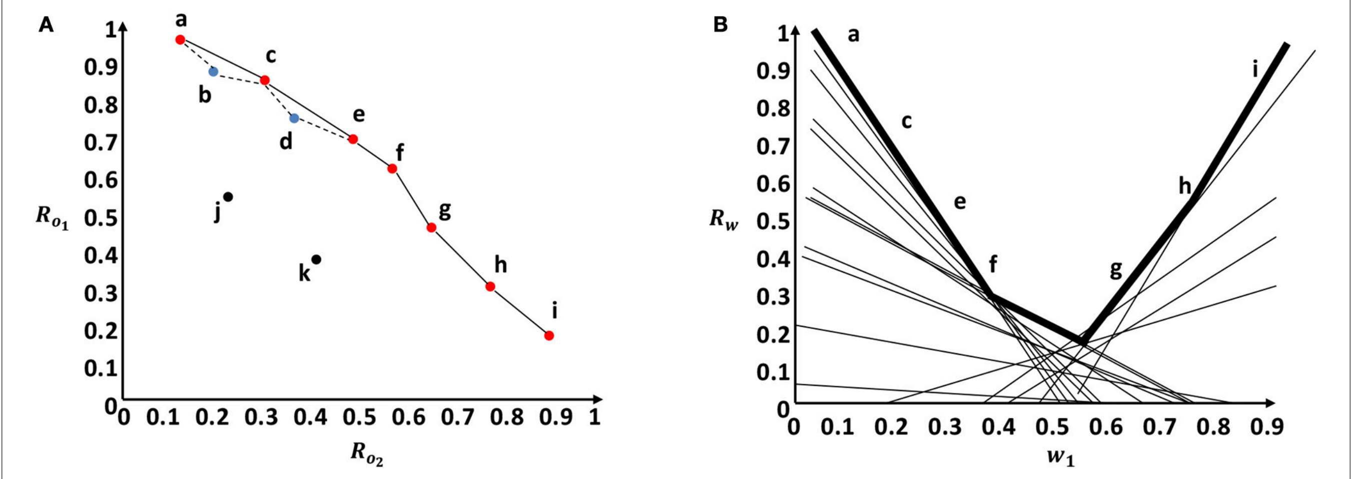


FIGURE 3 | Graphical representation of the Convex hull concept in comparison to the Pareto front using a two objective example. **(A)** Pareto front surface represented by solid and dotted lines vs. Convex hull surface represented only by solid lines. **(B)** Convex hull surface in the weight space represented by the bold lines.

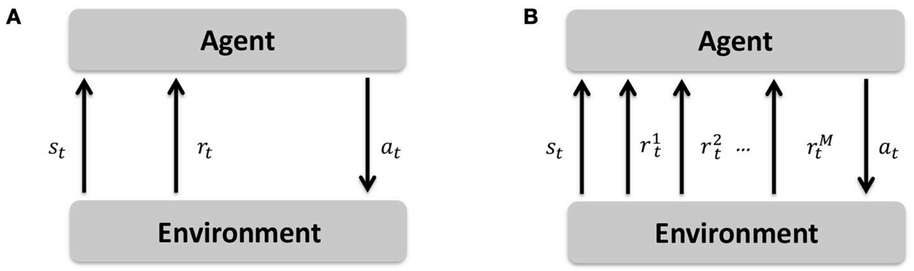


FIGURE 4 | Markov decision process (MDP) in comparison to multi-objective Markov decision process (MOMDP). **(A)** Markov decision process (MDP). **(B)** Multi-objective Markov decision process (MOMDP).

scalarized reward return for any given set of preferences within a *T time horizon*:

$$\max\ R_t^{\vec{w}^i} = \mathbb{E}[\sum_{j=0}^{T} \gamma^j h(\vec{r}_{t+j+1}, \vec{w}^i)] \tag{3}$$

$$\min\ |CCS|$$

$$s.t.\ \vec{w}^i \in W\ \forall \vec{w}^i \in \mathbb{R}^M, \sum_{m=1}^{M} w^m = 1$$

Where W is the set of all legitimate user's preferences over the defined objectives.

3. RELATED WORK

In this section, we explore the related work for multi-objective reinforcement learning (MORL) and intrinsically motivated reinforcement learning (IMRL), to highlight the contribution of our paper.

3.1. Multi-Objective Reinforcement Learning (MORL)

MORL methods address the MOMDP problem by two main approaches: single policy approaches; and multiple policy approaches (Roijers et al., 2013). If the user's preference is known before solving the problem, then a single policy can be found by scalarizing the multiple reward signals and optimizing the scalarized reward return using conventional single objective reinforcement learning methods. However, this assumption is rarely satisfied. Alternatively, the multiple policy approach aims at exploring and ranking the non-dominated policies in order to find the policy coverage set that can satisfy any user's preference for solving the problem. In the following subsections, we review relevant literature for each of these two approaches.

3.1.1. Single Policy Approaches

Lizotte et al. (2010) proposed a value iteration algorithm for ranking actions in finite state spaces using a linear scalarization function. Moffaert et al. (2013) proposed an updated version of the Q-learning algorithm (Watkins and Dayan, 1992) using the Chebyshev scalarization function to solve an MOMDP grid-world problem. Castelletti et al. (2013) utilized non-linear scalarization methods with a random weight space exploration technique to optimize the operation of water resource management systems. Perny and Weng (2010) addressed the MOMDP problem using a linear programming technique adopting the Chebyshev scalarization function. Ogryczak et al. (2011) extended previously mentioned linear programming method by replacing the non-linear scalarization with an ordered weighted regret technique for ranking actions. Their technique estimates the regret value per each objective with respect to a reference point, then actions are ranked using the combined regret value overall objectives.

Alternatively to the scalarization approach, constrained methods for the MOMDP problem have been introduced by Feinberg and Shwartz (1995) and Altman (1999). These methods optimize a single objective, while treating the other objectives as constraints on the optimization problem.

3.1.2. Multiple Policy Approaches

A preference elicitation approach has been proposed by Akrour et al. (2011) to incorporate an expert's preference during the policy learning process in an algorithm called preference-based policy learning (PPL). Basically, the proposed algorithm needs a parameterized formalism of the policy in order to sample different trajectories by sampling from the parameter space, then the expert provides his qualitative preference based on the lately demonstrated trajectories, which is used to optimize the policy's parameters in a way that maximizes the expert's expected feedback. Similarly, Fürnkranz et al. (2012) proposed a framework for ranking policy trajectories based on qualitative feedback provided by the user. However, this methodology requires reaching the Pareto front of optimal policies in the beginning, then ranking trajectories samples from those policies according to the user's feedback.

An evolutionary computation method was introduced by Busa-Fekete et al. (2014) in order to generate the set of non-dominated policies shaping the Pareto front. Then, at each state, they rollout actions from this Pareto optimal set and rank them given the user's feedback in order to identify the optimal action to follow.

Roijers et al. (2014) proposed the Optimistic Linear Support (OLS) algorithm which aims at evolving an approximate policy coverage set by examining different possible weight vectors of the defined objectives. For example, if there are two objectives in the problem, it starts by examining the two corner preferences (i.e., $[0.1, 0.9]$, $[0.9, 0.1]$) and evolves two optimal policies for those preferences through single-objective reinforcement learner (i.e., Q-learning). Then, the algorithm is going to evaluate the performance of the two evolved policies in terms of average reward achieved given a threshold value (epsilon). The policy that will exceed this value will be added to the coverage set. Afterwards, the algorithm will try to find a mid-point preference between each explored preference pairs and repeat the performance evaluation against the defined threshold until no more performance enhancements are achieved.

Gábor et al. (1998) introduced the Threshold Lexicographic Ordering (TLO) algorithm which starts with a sample of uniformly distributed preferences and for each of them it evolves a policy by selecting at each state one of the optimal actions (each dedicated with a specific single objective given its weight) exceeding a threshold value or taking the action with the max value if all actions are below the threshold value. Similarly, the decision to add a policy to the coverage set is made given a specific performance threshold value.

The two latter algorithms have been used in many of MORL literature (Geibel, 2006; Roijers et al., 2015; Mossalam et al., 2016) to find a coverage set of policies that solves the MOMDP problem. It has to be noted that both of these algorithms follow an iterative preference exploration approach that require simulation on the environment assuming stationary dynamics in order to evolve the policy coverage set. However, our proposed method aims at evolving such coverage set in a developmental and adaptive manner with stationary and non-stationary environment's dynamics.

3.2. Intrinsically Motivated Reinforcement Learning (IMRL)

Inspired by the learning paradigms in humans and animals, computational models for intrinsically motivated learning aim at learning guided by internally generated reward signals. Ryan and Deci (2000) defined intrinsic motivation as performing activities for their inherit satisfaction instead of separable consequences. They further explained that this is similar to humans performing actions for fun or challenge rather than being directed to perform it due to external pressure or rewards. Intrinsically motivated reinforcement learning (IMRL) aims at extending the conventional reinforcement learning paradigm by allowing the learner agent to generate an intrinsic reward signal that either can supplement the extrinsic reward signal or completely replace it (Barto, 2013). Basically, this intrinsic reward signal can provide assistance to the learning agent when dealing with a sparse extrinsic reward signal, enhance the exploration strategy, or completely guides it to achieve the task.

There are multiple drives to the intrinsic motivation in literature such as curiosity, novelty, happiness, emotions, or surprise (Singh et al., 2009). Despite of the differences between their fitness functions, they are positioned around the same assumption that the learning agent only needs to use its internal and external state representations in order to calculate the intrinsic reward signal. Therefore, the agent can generate such a reward independent of external (task-specific) reward signals. Schmidhuber (2010) describes the learning assumption of IMRL as "maximizing the fun or internal joy for the discovery or creation of novel patterns." According to his perspective, a pattern is a sequence of observed data that is compressible. Compression here means that an encoding program can find a compact representation of the data sequence that is sufficient to regenerate the original sequence or predict any occurrence within it given the predecessor occurrences (Ming and Vitányi, 1997). While the novelty of the pattern means that the learning agent initially did not expect it but it could learn it. The pattern discovering/creation progress can be projected into an intrinsic reward for a conventional RL algorithm that acts to optimize it and consequently encouraging the agent to discover/create more novel patterns.

IMRL methods can be categorized differently based on either a reward source perspective or an objective perspective. For the reward source perspective categorization, Merrick and Maher (2009) indicated that IMRL methods can fall into two broad categories: methods that use both extrinsic and intrinsic reward signals; and methods that use only intrinsic reward signals. Alternatively, Oudeyer and Kaplan (2009) proposed a different categorization from an objective perspective. They divided the IMRL literature into three main groups based on the objective of the intrinsic motivation learning process: knowledge-based models, competency-based models, and morphological models. We adopt a knowledge-based intrinsic motivation model according to the objective categorization that falls into the first category of the reward source perspective as it used both extrinsic and intrinsic reward signals. Accordingly, we only explore knowledge-based intrinsic motivation relevant literature in this paper.

One of the early approaches to knowledge-based IMRL was proposed by Schmidhuber (1991b) which included two recurrent neural networks (RNNs): a model network, and a control network. The model network aimed at learning to model environmental dynamics in terms of predicting the state transitions conditioned on action taken. While the control network optimizes the action selection policy to explore states space regions in which the model network has high marginal uncertainty (prediction error). The control network is guided by intrinsic reward represented by the model network's prediction error. This method is considered a category II as it works mainly with intrinsic reward signals.

Pathak et al. (2017) proposed an intrinsically motivated exploration technique following a predictive perspective. They indicated two main objectives for the proposed technique. First, to learn representative features that distill the state-space features that are controllable by the agent capabilities from those that are out of the agent's control. Then, using these learned representative features, they optimize a predictive model for the state transition probability distribution. In order to achieve the first objective, an inverse dynamics model was used to learn the action taken based on the encoding (features) of the states before and after taking the action, using the experience replay buffer. The authors stated that this inverse dynamics inference technique will discourage learning encodings (features) that cannot affect or being affected by the agent's actions. While for the second objective, a forward dynamics model was proposed to predict the next state encoding based on the current state encoding and the action taken. The intrinsic reward was formulated as the prediction error of the forward dynamics model and combined with the extrinsic reward using summation. The learning agent uses the combined version of the extrinsic and intrinsic rewards to optimize the current policy.

Qureshi et al. (2018) targeted robotics domains for the application of intrinsic motivation. The authors proposed an intrinsically motivated learning algorithm for a humanoid robot

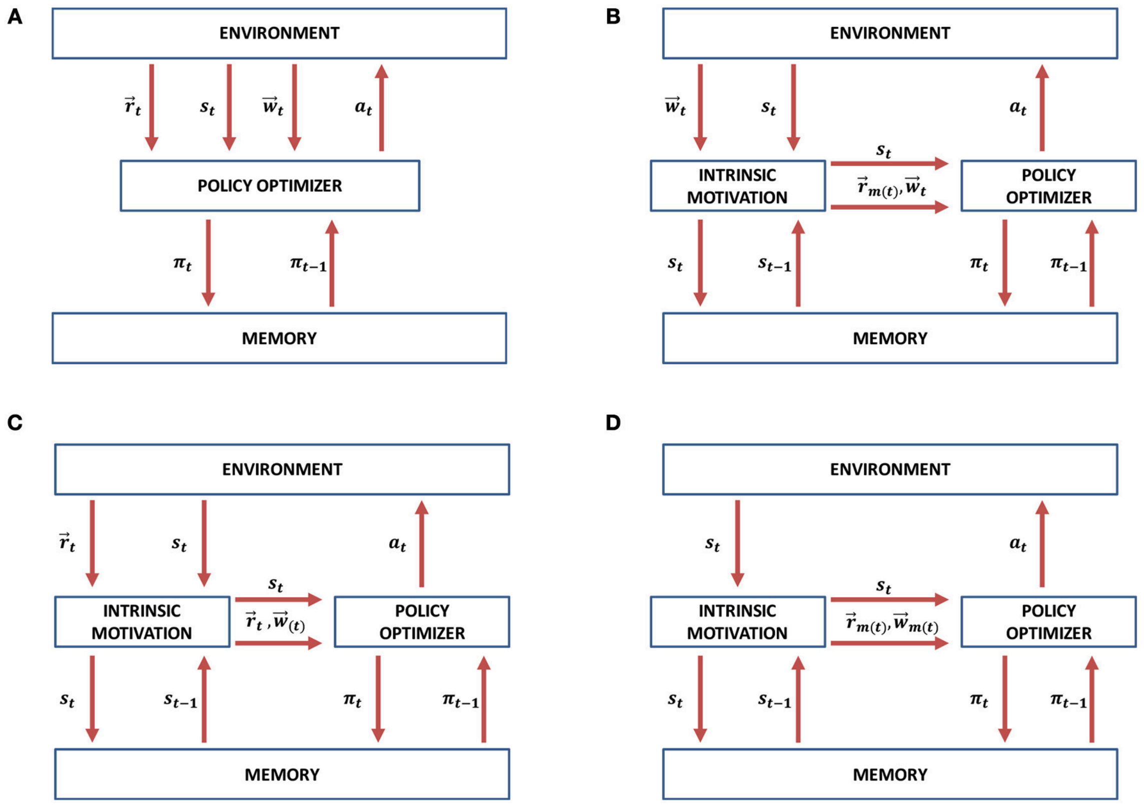


FIGURE 5 | Intrinsically motivated multi-objective reinforcement learning (IM-MORL) design scenarios. **(A)** The conventional MORL approach. **(B)** IM-MORL design approach guided by the user's preference. **(C)** IM-MORL design approach for learning the feasible preferences over multiple extrinsic rewards. **(D)** IM-MORL design approach for learning both internal goals and preferences.

to interact with a human given three basic events to represent the current state of the interaction: eye contact, smile, and handshake. Their algorithm is based on an event predictive objective where a predictive neural network called Pnet is learning to predict the coming event conditioned on the current one and the action taken, while another controller network called Qnet is optimizing the action selection policy guided only with the intrinsic reward represented by the prediction error of the Pnet. The authors showed that their proposed algorithm outperformed a conventional reinforcement learning algorithm using only extrinsic sparse reward signal in a real interaction experiment with humans that lasted for 14 days.

One drawback of formulating the intrinsic reward based on prediction error is that it encourages the action sampler (e.g., control network) to favor state space regions that involve noisy observation or require further sensing capabilities beyond the currently available to the agent, this might limit the learning progress of the whole system in such situations.

In order to overcome this drawback, we need to change the formulation of the intrinsic reward to depend on the model's improvement (e.g., prediction accuracy) rather than its prediction error. Consequently, the learning agent will be bored from state-space regions that either completely predictable (high prediction accuracy) or completely unpredictable (due to noise or lack of sufficient sensors) as for both scenarios the gradient of the improvement will be small.

A first attempt to tackle this issue was proposed by Schmidhuber (1991a), where the intrinsic reward was formulated based on prediction reliability rather than the error. A probabilistic inference model was optimized to learn the state transition probability distribution conditioning on the taken action, then four different metrics were proposed to estimate the prediction reliability locally and globally based on the past interactions with the environment. A Q-learning algorithm was adopted to optimize the action selection policy guided by the reliability value as an intrinsic reward signal. The proposed methodology was evaluated on a non-deterministic environment with noisy state regions and compared with a random-search exploration technique, results showed that the intrinsically motivated agent was 10 times faster to decrease the prediction error.

Oudeyer et al. (2007) proposed a developmental learning system for robotics called intelligent adaptive curiosity (IAC). The IAC system aims at maximizing the learning progress of the agent represented by focusing the learning process on situations that neither fully predictable nor fully unpredictable, as the derivative of the progress will be small in both situations. The novelty in this method comes in the division of the state space into regions that share common dynamics and for each region, the IAC evolves an expert predictive model (e.g., neural network) to learn the state transition dynamics. The division of the state space into regions was done in a developmental manner, so at the beginning, there is only one region and when the number of examples exceeds a specific threshold value (C_1) then it is split into two regions based on a second metric (C_2) that aims at minimizing the variance between samples in a specific region (i.e., this is symmetric to density-based clustering techniques Kriegel et al., 2011). The intrinsic reward is calculated using the first derivative of the prediction error between times (t and $t + 1$). Finally, a Q-learning algorithm is adopted to optimize the action selection policy guided by the intrinsic reward. The authors showed by experiments the effectiveness of the proposed system in comparison to conventional exploration strategies.

Our propose intrinsically motivated preference exploration component follows the same intrinsic reward formulation approach as the last two methods based on the predictive model improvement rather than the prediction error. However, we extend the existing work to multi-objective scenarios.

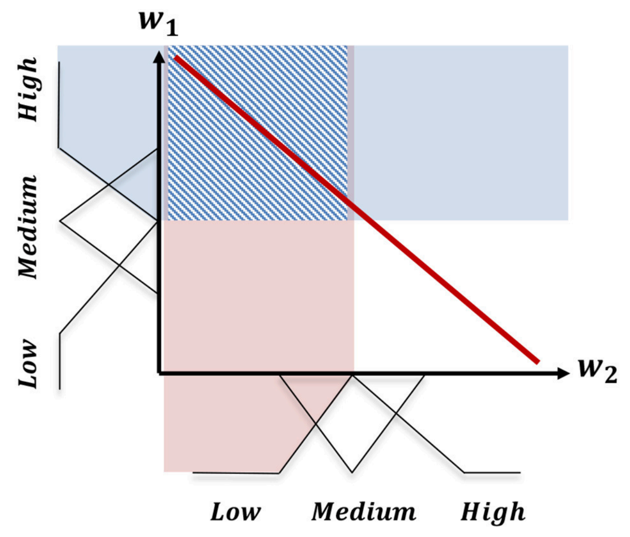


FIGURE 6 | The division of the linear scalarization of the preference space into a finite set of regions based on the combination of fuzzy membership values of the weight components.

4. METHODS

In comparison to the conventional MORL approach presented in **Figure 5A**, we propose three possible scenarios in which intrinsically motivated multi-objective reinforcement learning (IM-MORL) approaches can be designed. In the first scenario, the user can supply his/her preference over a defined set of intrinsic motivation rewards, while the intrinsic motivation system can utilize this preference to formulate a combined intrinsic reward to guide the learning agent according to the user's preference (see **Figure 5B**). An example of this scenario is for a child

TABLE 1 | Configuration of the utilized triangular fuzzy membership functions.

Function	A	B	C
Low	0.00	0.18	0.35
Medium	0.28	0.45	0.65
High	0.57	0.75	1.00

that has multiple intrinsic motives, while his/her parent is guiding his/her behavior by providing feedback that can form an acceptable trade-off among these internal motives. While in the second scenario, the environment will supply a vector of extrinsic rewards and the task of the intrinsic motivation system will be to learn the feasible preferences that could solve the task and evolve a policy for each of them (see **Figure 5C**). An example of this scenario is for a student who is given curricula of learning courses and his/her mission is to find an optimal strategy to maximize the total grade among all of them. Finally, the intrinsic motivation can generate both the rewards and preferences completely internally without depending on any external source for each of them (see **Figure 5D**). This scenario is similar to a human adult who is behaving in a free-willed manner in order to learn a set of internally generated goals, while evolving his/her prioritization among

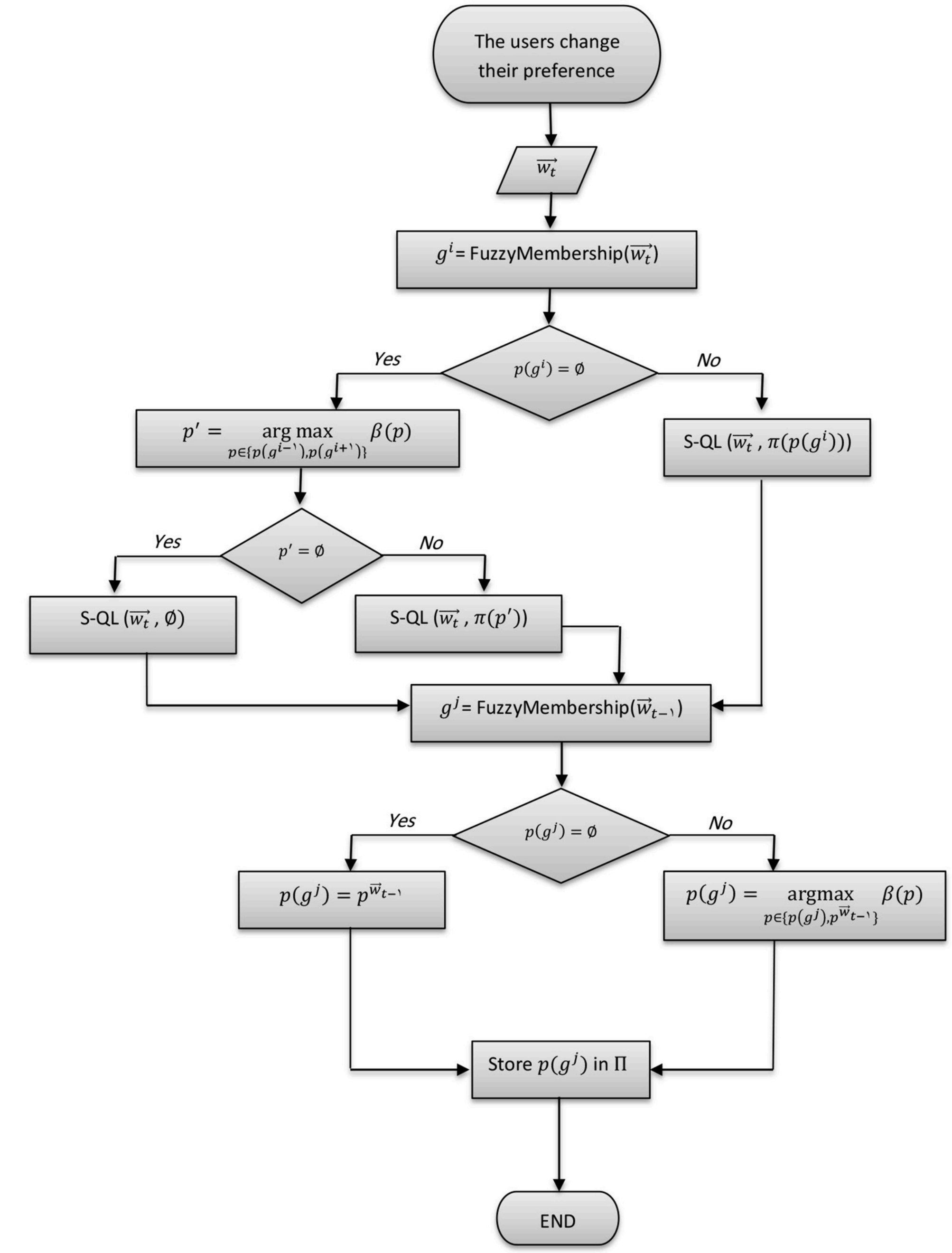

The users change their preference
$\overrightarrow{w_t}$
g^i= FuzzyMembership($\overrightarrow{w_t}$)
$p(g^i) = \emptyset$
Yes
No
$p' = \arg\max_{p \in \{p(g^{i-1}), p(g^{i+1})\}} \beta(p)$
S-QL ($\overrightarrow{w_t}$, $\pi(p(g^i))$)
$p' = \emptyset$
Yes
No
S-QL ($\overrightarrow{w_t}$, $\emptyset$)
S-QL ($\overrightarrow{w_t}$, $\pi(p')$)
g^j= FuzzyMembership($\overrightarrow{w}_{t-1}$)
$p(g^j) = \emptyset$
Yes
No
$p(g^j) = p^{\overrightarrow{w}_{t-1}}$
$p(g^j) = \arg\max_{p \in \{p(g^j), p^{\overrightarrow{w}_{t-1}}\}} \beta(p)$
Store $p(g^j)$ in Π
END

FIGURE 7 | A flowchart diagram describing the RFPB algorithm workflow.

them according to his/her current achievement level on each goal.

Our proposed IM-MORL method follows the second approach depicted in **Figure 5C**. We leave the other approaches for future exploration. In this paper, the agent gets extrinsic rewards from the environment and automatically explores the preference space in order to evolve the optimum policy coverage set that solves the MOMDP problem. This is achieved through adversarial self-play between two main components: the intrinsically motivated preference explorer; and the convex coverage set optimizer. The former component explores preferences for which there is no an optimum policy in the CCS, while the latter component optimizes policies that can maximize the scalarized reward return for preferences proposed by the former component. Consequently, through this adversarial interaction, the proposed method developmentally evolves the CCS that converges to the optimal CCS to solve the problem. We are going to describe each of these components in details as follows.

4.1. Convex Coverage Set Optimizer

In order to respond to preferences proposed by the preference exploration component, we propose a novel convex coverage set optimization algorithm called robust fuzzy policy bootstrapping (RFPB). The main assumption of the RFPB algorithm is *While there is a large number of policies that can satisfy different preferences over the defined objectives, a fewer number of steppingstone policies can be used to solve the problem by bootstrapping specialized policies that can fit any feasible preference*. The concept of policy bootstrapping from steppingstone policies achieves better robustness to changes in the environment setup in comparison to greedy policies optimized for a specific setup or user's preference.

The RFPB algorithm divides the linear scalarization of the preference space into a finite number of regions each is dedicated to a specific combination of fuzzy membership values for the weight components in the preference. The advantage of using fuzzy representation instead of alternative heuristic discretization methods is that it enables automatic categorization of the preference regions in terms of combinations of different fuzzy membership functions without the need to tailor specific rules for such categorization in the crisp representation case.

For further explanation of this fuzzy representation, consider the example in **Figure 6**. In this example, there are two defined objectives: o_1; and o_2. Accordingly, the user's preference can be defined as a two-dimensional vector $\vec{w}_i = [w_1, w_2]$, $\vec{w}_i \in \mathbb{R}^2$ defining a tradeoff across these two objectives. If we define three triangular membership functions (low, medium, and high) for each weight component in the preference, we will end up with (3×3) nine combinations of membership values. Consequently, the convex hull can be represented using these nine regions of the weight space membership values. The shaded square in **Figure 6** represents the region for the fuzzy membership combination $w_1 = High$, and $w_2 = Low$.

In this paper, we use the triangular fuzzy membership functions (Zadeh, 1996). Thus, there are three fuzzy membership functions for each weight component including low, medium, and high functions. The configuration of these sets is presented in **Table 1**. Each combination of these fuzzy membership functions gives a fuzzy preference region. As the weight components are constrained to sum to one (Definition 2.3), the extreme regions (low,low or high, high in the example presented in **Figure 6**) are excluded from the set of legitimate regions.

After defining this fuzzy regions, the RFPB algorithm evolves a single steppingstone policy for each region. A policy (p^g) is assigned to the fuzzy region (g) if there is no other policy that dominates (p^g) on the robustness metric (β^g) for the region (g). In this paper, we use the robustness metric defined in Equation (4):

$$\beta^k = \frac{\Gamma^k}{\sigma^k} \tag{4}$$

The logic behind this metric is that it calculates the robustness of a policy (p^k) as a tradeoff between its performance represented by it average reward value (Γ^k) and its variability represented by its standard deviation value (σ^k). Therefore, this metric favors stable policies that can serve as steppingstones to evolve specialized policies within its fuzzy preference region. The robustness metric utilizes the average and standard deviation of the values generated by the scalarized reward function presented in Equation (2) during the time period from deploying the policy to the time of the preference region change. Moreover, this metric is related to the problem definition in section 2.3 through assuring the robustness of the steppingstone policy assigned for each preference region, therefore, the performance

TABLE 2 | Parameters configuration for the DNN predictive model.

Parameter	Value
Layers	Sigmoid(3), ReLU(32), ReLU(16), ReLU(8), Linear(1)
α	0.09
Dropout	0.3
Cost function	Cross entropy
Optimizer	ADAM

TABLE 3 | Parameters configuration for the utilized DDPG algorithm in the exploration component.

Parameter	Value
τ	0.001
γ	0.99
Actor α	0.0001
Critic α	0.001
Ornstein-Uhlenbeck Noise θ	0.15
Ornstein-Uhlenbeck Noise σ	0.2
Optimizer	ADAM

Algorithm 1 Scalarized Q-Learning (S-QL)

```
Input: A preference $\vec{w}$.
1: if $\pi^{init} = \phi$ then
2:     Initialize $Q(s,a)\ \forall s \in S,\ a \in A(s)$ arbitrarily
3: else
4:     Initialize $Q(s,a)\ \forall s \in S,\ a \in A(s)$ from $\pi^{init}$
5: repeat
6:     for each episode do
7:         Initialize $S$
8:         Take $a$ from $s$ using policy derived from $Q$ (e.g.,
9:         $\epsilon$-greedy) , observe $\vec{r},\ s'$
10:        Calculate scalarized reward $\rho = \vec{w} \cdot \vec{r}$
11:        $Q(s,a) \leftarrow Q(s,a) + \alpha\big[\rho + \gamma max_{a'} Q(s',a') - Q(s,a)\big]$
12:        $s \leftarrow s'$
13: until $s$ is terminal
```

Algorithm 2 Robust Fuzzy Policy Bootstrapping (RFPB)

```
Input: Preferences at times $t$ and $t-1$ ($\overrightarrow{w}_t$, $\overrightarrow{w}_{t-1}$).
1: Get the fuzzy region of the new preference
   $FuzzyMembership(\overrightarrow{w}_t) \rightarrow g^i$
2: if $p(g^i) \neq \emptyset$ then
3:     $p' := p(g^i)$
4: else if $p(g^{i-1}) \neq \emptyset$ and $p(g^{i+1}) \neq \emptyset$ then
5:     $p' := \arg\max_{p \in \{p(g^{i-1}),\, p(g^{i+1})\}} \beta(p)$
6: else if $p(g^{i-1}) = \emptyset$ and $p(g^{i+1}) = \emptyset$ then
7:     $p' := \phi$
8: else
9:     $p' := \arg_{p \in \{p(g^{i-1}),\, p(g^{i+1})\}} p \neq \phi$
10: Get the fuzzy region of the old preference
    $FuzzyMembership(\overrightarrow{w}_{t-1}) \rightarrow g^j$
11: if $p(g^j) \neq \emptyset$ then
12:     $p(g^j) := \arg\max_{p \in \{p(g^j),\, p^{w_{t-1}}\}} \beta(p)$
13: else
14:     $p(g^j) := p^{w_{t-1}}$
15: Store $p(g^j)$ in $\Pi$
16: if $p' = \emptyset$ then
17:     $\pi' := \phi$
18: else
19:     $\pi' := \pi(p')$
20: Follow the Scalarized Q-Learning algorithm, S-QL($\overrightarrow{w}_t$, $\pi'$)
```

overall legitimate preferences can be maximized as targeted in the objective function.

Our proposed methodology adopts a scalarized version of Q-learning for solving the MOMDP task using linear scalarization given the weights vector as depicted in Algorithm 1. We refer to this algorithm as Scalarized QL abbreviated as (S-QL).

As shown in Algorithm 2, when a new user's preference ($\vec{w}_t$) is introduced at the time (t), it is assigned a fuzzy representation based on the membership functions that have the maximum values for its weight components. Consequently, a region (g^i) is determined from the fuzzified representation of the convex hull corresponding to the new preference. The new policy will be bootstrapped from the non-dominated policy of the region (g^i). In the case that region (g^i) was not explored before, the new policy is bootstrapped from the policy that achieved the higher robustness value over adjacent regions (g^{i-1}) and (g^{i+1}). The two adjacent regions are determined by measuring the Euclidean distance (see Equation 5) between the centroids vectors (i.e., the b components of the corresponding triangular membership functions) of the current region and each of the remaining regions, then, taking the top two nearest regions. In the case that the adjacent regions were not explored, then the new policy is initialized arbitrarily. The policy for the last preference ($p^{w_{t-1}}$) is compared to the current non-dominated policy of its fuzzy region $p(g^j)$ based on the robustness metric (β). If it exceeds the non-dominated policy, then it will take its position in the policy repository (Π) for that region.

$$\mathrm{Euclidean\,Distance}(g^i, g^k) = \sqrt{\sum_{m=1}^{M} (g^i_{b_m} - g^k_{b_m})^2} \qquad (5)$$

The RFPB algorithm stores the past explored non-dominated policies over the preference fuzzy regions in a policy repository Π. As mentioned previously, a non-dominated policy p^k outperforms, in terms of the robustness metric (β^k), all explored policies within its k^{th} region. For each single non-dominated policy p^k, we store three basic parameters $\langle \pi^k, g^k, \beta^k \rangle$. Where $\pi^k \in \mathbb{R}^{N \times L}$ is the Q-value matrix for each state and action pair, g^k is the preference region assigned to the policy, and β^k is the robustness metric value calculated using Equation (4).

After bootstrapping, the RFPB algorithm will continue to optimize the policy with regard to the new preference region following the scalarized Q-learning (S-QL) algorithm depicted in Algorithm 1.

For further insights on the RFPB algorithm, **Figure 7** provides a flowchart diagram that describes the processes involved in its workflow.

4.2. Intrinsically Motivated Preference Exploration

This component adopts a knowledge-based intrinsic motivation approach (Oudeyer and Kaplan, 2009) to actively explore the preference space. Mainly, this component includes two building blocks. First, a predictive model, which is implemented as a deep feed-forward neural network (see **Table 2** for parameters configuration), is trained in a supervised learning manner to predicts the scalarized reward return (Equation 2) given a preference fuzzy region. The input to the predictor is the preference fuzzy region as one hot encoding vector (a binary valued vector with the length equal to the number of regions with only 1 value at the corresponding location of the current region), while the output is the predicted return to be achieved by the evolved CCS from the time of the preference proposal ($t-k$) to the end time of the policy execution ($t+j$). Before the beginning of the training process, there is a warming up period to accumulate training set of 200 samples for the predictor. During this period, preferences are proposed randomly (uniformly sampled) to the RFPB algorithm, which is given a maximum number of 100 episodes to evolve a corresponding policy and recording the resulting reward return at the end. Afterwards, the predictor is initialized based on this warm up data and the intrinsically motivated preference exploration is activated.

The second building block is responsible for the adaptive preference exploration. Mainly, it utilizes a reinforcement learning algorithm that observes the current preference fuzzy region as the state, takes an action with M dimensions representing the weight components for the defined objectives, and gets an intrinsic reward formulated as the difference (gain) in prediction accuracy (ρ) of the predictive model for the explored region (g) within the time period $[t-k, t+j]$, as per Equation (11). Basically, the reinforcement learning algorithm works as an active learning trainer to the predictive model and it is rewarded through maximizing the prediction accuracy gain after sampling a new interaction with the RFPB algorithm

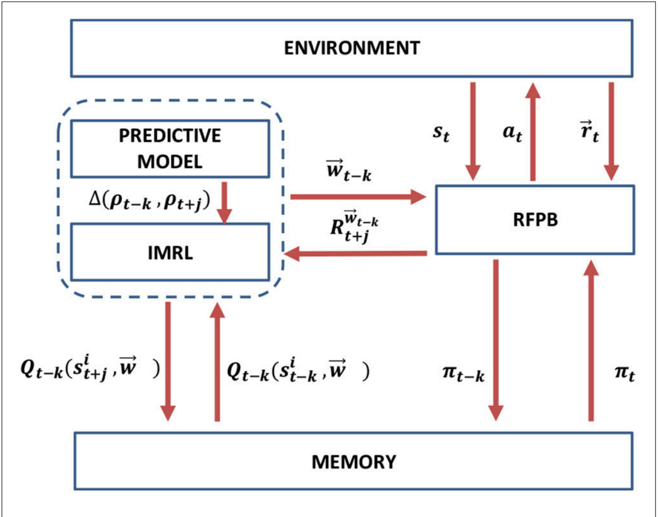

ENVIRONMENT
PREDICTIVE MODEL
$\Delta(\rho_{t-k}, \rho_{t+j})$
IMRL
$\vec{w}_{t-k}$
$R^{\vec{w}_{t-k}}_{t+j}$
s_t
a_t
$\vec{r}_t$
RFPB
$Q_{t-k}(s^i_{t+j}, \vec{w})$
$Q_{t-k}(s^i_{t-k}, \vec{w})$
π_{t-k}
π_t
MEMORY

FIGURE 8 | A block diagram for the working mechanism of the proposed method.

formulated as a tuple of (preference region, scalarized reward return) and adding it to the training set of the predictive model.

We utilized the deep deterministic policy gradient algorithm (DDPG) as described in Lillicrap et al. (2016) for the implementation of the reinforcement learning algorithm. The implementation configuration for the DDPG algorithm is presented in **Table 3**. The DDPG algorithm falls into the actor-critic reinforcement algorithms, therefore, there are two neural networks mainly involved in the learning process: the actor network (μ) which is responsible for taking actions, and the critic network (Q) which is responsible for estimating the Q-value of each state-action pairs. Using (N) number of transitions samples randomly from a previous transitions experience buffer, the critic aims at minimizing the loss function (L), while the actor is updated using the policy gradient ($\nabla_{\theta^\mu} J$).

$$L = \frac{1}{N}\sum_{n=1}^{N}(y_n - Q(s_n, a_n|\theta^Q))^2 \tag{6}$$

$$Where\ y_n = r_n + \gamma Q'(s_{n+1}, \mu'(s_{n+1}|\theta^{\mu'})|\theta^{Q'})$$

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_{n=1}^{N} \nabla_a Q(s, a\,|\theta^Q)\,|_{s=s_n, a=\mu(s_n)} \nabla_{\theta^\mu}\mu(s\,|\theta^\mu)\,|_{s_n} \tag{7}$$

In addition to these two main networks, the DDPG uses the concept of target networks (μ', Q'), which are basically replicas of the actor and critic networks but with an older version of the parameters (weight) configuration. The logic behind this is to enable stable learning by separating the network that is being optimized from the one that is performing the exploration. The parameters of the target networks are updated in proportional to their current values and latest values of the actor-critic networks using the τ parameter as follows:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{8}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{9}$$

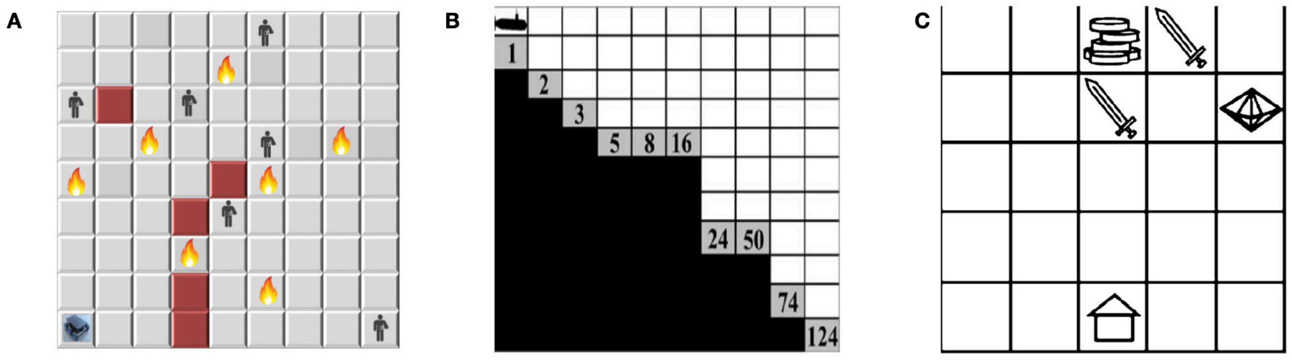


FIGURE 9 | Layouts of the experimental environments. **(A)** The search and rescue (SAR) environment. **(B)** The deep sea treasure (DST) environment. **(C)** The resource gathering (RG) environment.

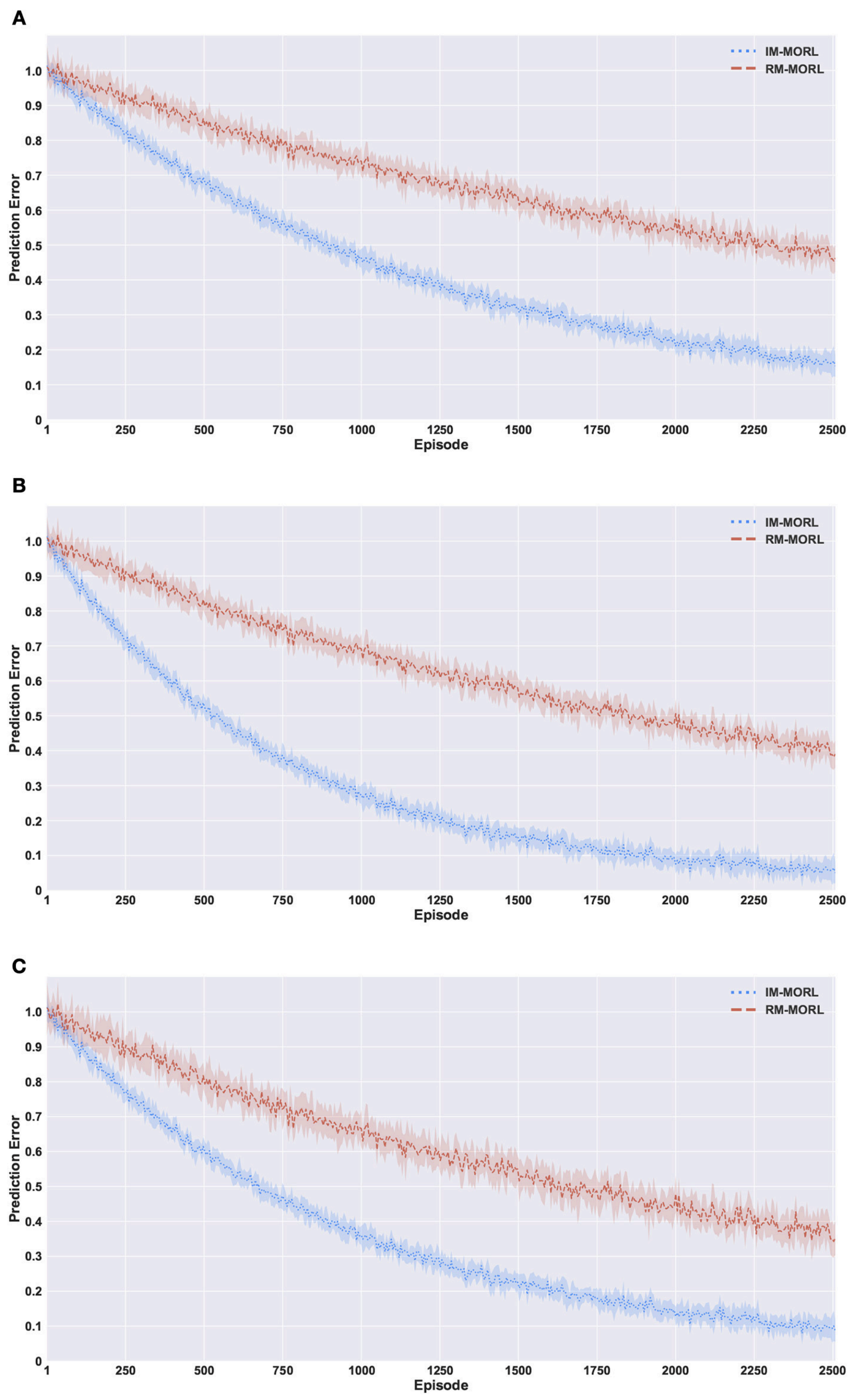


FIGURE 10 | Comparing our IM-MORL with the RM-MORL agent in terms of reward prediction error averaged over 15 runs to assess the impact of intrinsically motivated preference exploration. **(A)** The search and rescue (SAR) environment. **(B)** The deep sea treasure (DST) environment. **(C)** The resource gathering (RG) environment.

TABLE 4 | Comparison results between our IM-MORL agent and RM-MORL agent in terms of average prediction error with standard deviation in 100 percentage over 15 runs per each of the experimental environments.

Environment	IM-MORL	RM-MORL
SAR	**16.4 ± 3.7**	48.1 ± 4.2
DST	**7.6 ± 2.9**	38.7 ± 5.1
RG	**9.2 ± 5.3**	35.2 ± 6.3

Bold value indicates best results.

The actions are explored using the OrnsteinUhlenbeck stochastic process, which generates temporally correlated exploration noise to make smooth transitions between action values. Accordingly, the preference exploration moves smoothly from one preference region to its adjacent regions, while exploring the preference space. The equation for this process is as follows:

$$da_t = \theta(\mu - a_t)dt + \sigma dW_t \tag{10}$$

Where θ, σ, and μ are parameters and W_t represents the Wiener process.

$$r^{intrinsic} = \Delta(\rho^g_t, \rho^g_{t+k}) \tag{11}$$

Figure 8 presents a block diagram for the interaction between the two described components of the proposed method.

5. EXPERIMENTAL DESIGN

In this section, we describe our experimental design to evaluate the proposed method.

5.1. Experiments

5.1.1. Assessing the Impact of the Intrinsically Motivated Preference Exploration

Aim: The aim of this experiment is to assess the impact of the intrinsically motivated preference exploration on the performance of the reward prediction model, which reflects the stability of the CCS performance.

Workflow: We compare our proposed intrinsically motived agent with a randomly motivated agent that samples preference uniformly from the M-dimensional weight space to train both the reward predictive model and the CCS optimizer. We execute 15 runs, each one lasts for 2,500 episodes per each experimental environment. We refer to our proposed agent as IM-MORL and the randomly motivated agent as RM-MORL.

Evaluation Criteria: We calculate the average and standard deviation of the prediction error of the reward predictive model over the 15 runs, each with a different environment setup (different distribution of objects), per each experimental environment. The less the reward prediction error value, the more effective the preference exploration strategy and the more stable the performance of the resultant CCS.

5.1.2. Comparison to the State-of-the-Art MORL Algorithms

Aim: This experiment aims at contrasting the performance of our IM-MORL method with the state-of-the-art MORL methods under both stationary and non-stationary environments.

Workflow: We compare our method with two well-known and highly adopted methods in MORL literature (as described in section 3): the Optimistic Linear Support (OLS) method (Roijers et al., 2014); and the Threshold Lexicographic Ordering (TLO) method (Gábor et al., 1998). We conduct this experiment in two environment groups: stationary environments; and non-stationary environments. In the former, the distribution of objects in the environment is stationary per each run. While in the latter, the distribution of objects is non-stationary as 25% of them change their locations randomly every 100 episodes. For each group, we execute 15 runs that differ in the distribution of objects in the experimental environment. Each run is divided into a training phase and a testing phase, each of them includes 2,500 episodes. The training phase allows each method to evolve its CCS. While in the testing phase, we sample ten user preferences uniformly, and every 250 episodes the preference changes to evaluate the performance of the evolved CCS for each method. Moreover, during the testing phase, the exploration component in our proposed method is inactive, while the RFPB algorithm keeps updating the CCS by replacing inferior steppingstone policies with better ones based on the robustness metric defined in Equation (4). For the parameter configuration of the OLS and TLO algorithms we follow the same configuration in Roijers et al. (2014) and Geibel (2006) respectively.

Evaluation Criteria: We evaluate the three comparative methods over two main metrics. First, the sum of median rewards metric, which is calculated by taking the median reward value for each preference, sum them for each run, then taking the average of this sum over the 15 runs. This metric reflects the overall performance of the evolved CCS for each comparative method over the 15 independent runs executed. For visualizing this evaluation, we show the average median value with standard deviation for each sampled preference. Second, the *hypervolume* metric which measures the coverage and diversity of the CCS. The higher the value of this metric the better the CCS. We followed the algorithm described in Beume et al. (2009) to calculate the value of this metric.

5.2. Environments

We use three different multi-objective sequential decision-making environments: search and rescue; deep-sea treasure; and resources gathering. The two later environments are well-known benchmarks in the MORL literature (Vamplew et al., 2011), while the first environment is a new and firstly proposed in this paper. The proposed environment poses an additional challenge of stochastic state transition distribution. **Figure 9** shows the layout of the experimental environments.

5.2.1. Search and Rescue (SAR) Environment

This 9 × 9 grid world represents a SAR scenario that has fire danger, obstacles, and human victims to be rescued. The agent's state is a tuple $\langle X, Y, F, O, H\rangle$, where X, Y are the coordinates of

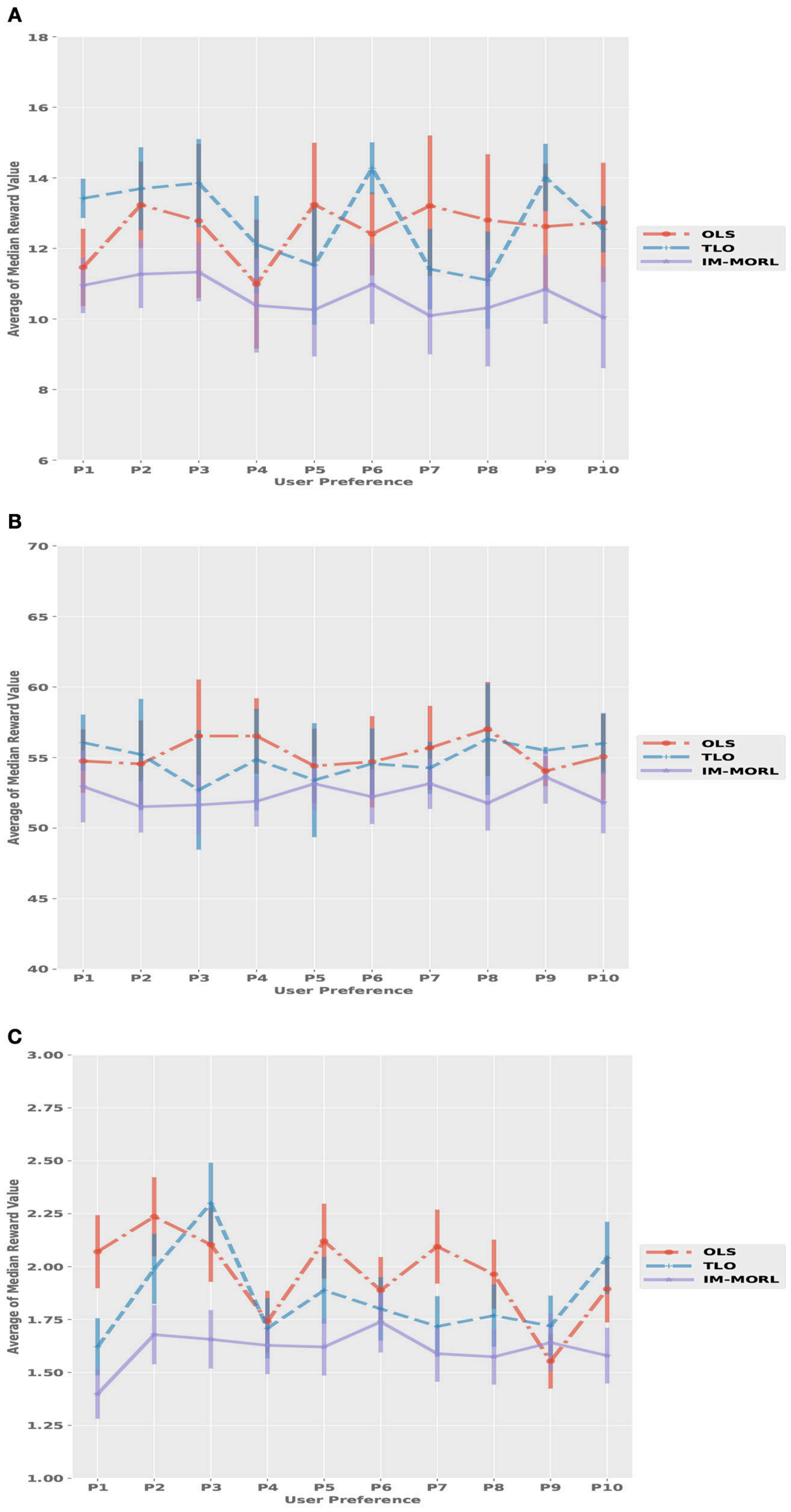


FIGURE 11 | Comparing the median reward values for each user preference averaged over 15 runs with standard deviation bars in the stationary environments. **(A)** The search and rescue (SAR) environment. **(B)** The deep sea treasure (DST) environment. **(C)** The resource gathering (RG) environment.

the current location, and F, O, H are binary values indicating whether a fire danger, an obstacle, or a human victim is in the current location or not. Moving to an obstacle won't change the location while getting a time penalty. Each human victim die after a random time ξ_i, $i \in \{1, 2, 3, \ldots, N\}$ for N victims. The action space is $A = \{MoveEast, MoveWest, MoveNorth, MoveSouth\}$ with one square per each movement. There are three objectives in this environment: maximizing the number of detected human victims; minimizing exposure to fire risk; and minimizing the overall task's time. The agent gets a vector of three rewards $\vec{r} = \left[r^{victim}, r^{fire}, r^{time}\right]$, $\vec{r} \in \mathbb{R}^3$. The victim reward function r^{victim} is $+3$ for each detected victim and 0 elsewhere, the fire penalty function r^{fire} is -5 for each exposure and 0 elsewhere, and the time penalty function r^{time} is always set to -1.

5.2.2. Deep Sea Treasure (DST) Environment

This is a 10×11 grid world. The agent controls a submarine searching for an undersea treasure. The agent's state is a tuple of$\langle X, Y\rangle$, where X, Y are the coordinates of the current position. There are four actions to move one square per each direction $A = \{Left, Right, Up, Down\}$. All actions that result in leaving the grid will not change the submarine's position. Multiple treasures can be found in this environment each with a different reward value. It has two objectives. First, to minimize needed time to find the treasure. Second, to maximize the treasure's value. Accordingly, the reward vector has two rewards $\vec{r} = \left[r^{time}, r^{treasure}\right]$, $\vec{r} \in \mathbb{R}^2$, where r^{time} is a time penalty of -1 on all turns and $r^{treasure}$ is the captured treasure reward which depends on the treasure's value.

5.2.3. Resources Gathering (RG) Environment

In this 5×5 grid world, the task is to collect resources (gold and gems) and return home. The agent's state is a tuple $\langle X, Y, G, Y, E\rangle$, where X, Y are the coordinates of the current location, and G, Y, E are binary values indicating whether a gold resource, a gem resource, or an enemy is in the current location or not. The enemy attack may occur with a 10% probability. If an attack happens, the agent loses any resources currently being carried and is returned to the home location. The action space is $A = \{MoveEast, MoveWest, MoveNorth, MoveSouth\}$ with one square per each movement. The objectives are to maximize the resources gathered while minimizing enemy attacks. The rewards vector is defined as $\vec{r} = [r^{resources}, r^{enemy}]$, $\vec{r} \in \mathbb{R}^2$, with $r^{resources}$ is $+1$ for each resource collected and r^{enemy} is -1 for each attack.

TABLE 5 | Comparing the OLS, TLO, and IM-MORL agents in terms of sum of median reward values averaged over 15 runs in the stationary environments.

Environment	OLS	TLO	IM-MORL
SAR	**125.3** $\pm$ **2.5**	124.7 $\pm$ 2.1	123.9 $\pm$ 4.5
DST	**539.4** $\pm$ **2.8**	536.7 $\pm$ 2.5	535.2 $\pm$ 4.5
RG	18.1 $\pm$ 2.8	**17.5** $\pm$ **1.8**	16.9 $\pm$ 1.2

Bold value indicates best results.

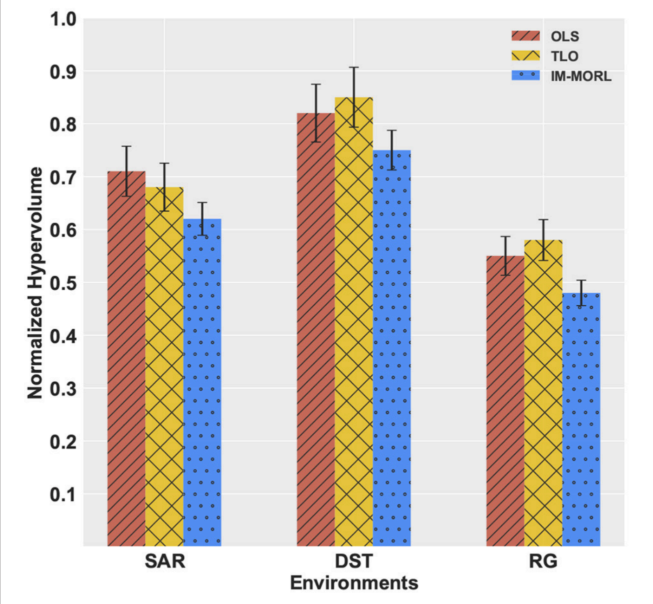


FIGURE 12 | A bar-chart comparing the normalized average *hypervolume* values with standard deviation for the OLS, TLO, and IM-MORL agents grouped by each stationary environment.

6. RESULTS AND DISCUSSION

In this section, we present and discuss the results of the two experiments defined in our experimental design.

6.1. Assessing the Impact of the Intrinsically Motivated Preference Exploration

Figure 10 presents the comparison results between our intrinsically motivated multi-objective reinforcement learning (IM-MORL) agent and a randomly motivated mullti-objective reinforcement learning (RM-MORL) agent, in order to assess the effectiveness of intrinsic motivation in preferences exploration. The results show the average prediction error over 15 runs for the DNN prediction model described in section 4.2, which aims at predicting the expected reward return per each preference fuzzy region given the current performance of the CCS.

Figure 10A shows the prediction error results in the search and rescue (SAR) environment. Our IM-MORL agent significantly outperformed the RM-MORL with 33% less error on average. **Figure 10B** shows the results in the deep sea treasure (DST) environment. Similarly, our IM-MORL agent significantly outperformed the RM-MORL with 21% less error on average. Finally, **Figure 10C** shows that our IM-MORL agent significantly outperformed the RM-MORL agent with 26% on average. We

TABLE 6 | Comparing the OLS, TLO, and IM-MORL agents in terms of average *hypervolume* over 15 runs in the stationary environments.

Environment	OLS	TLO	IM-MORL
SAR	**0.73** $\pm$ **0.05**	0.67 $\pm$ 0.05	0.65 $\pm$ 0.04
DST	0.82 $\pm$ 0.05	**0.85** $\pm$ **0.06**	0.75 $\pm$ 0.05
RG	0.55 $\pm$ 0.03	**0.58** $\pm$ **0.04**	0.48 $\pm$ 0.03

Bold value indicates best results.

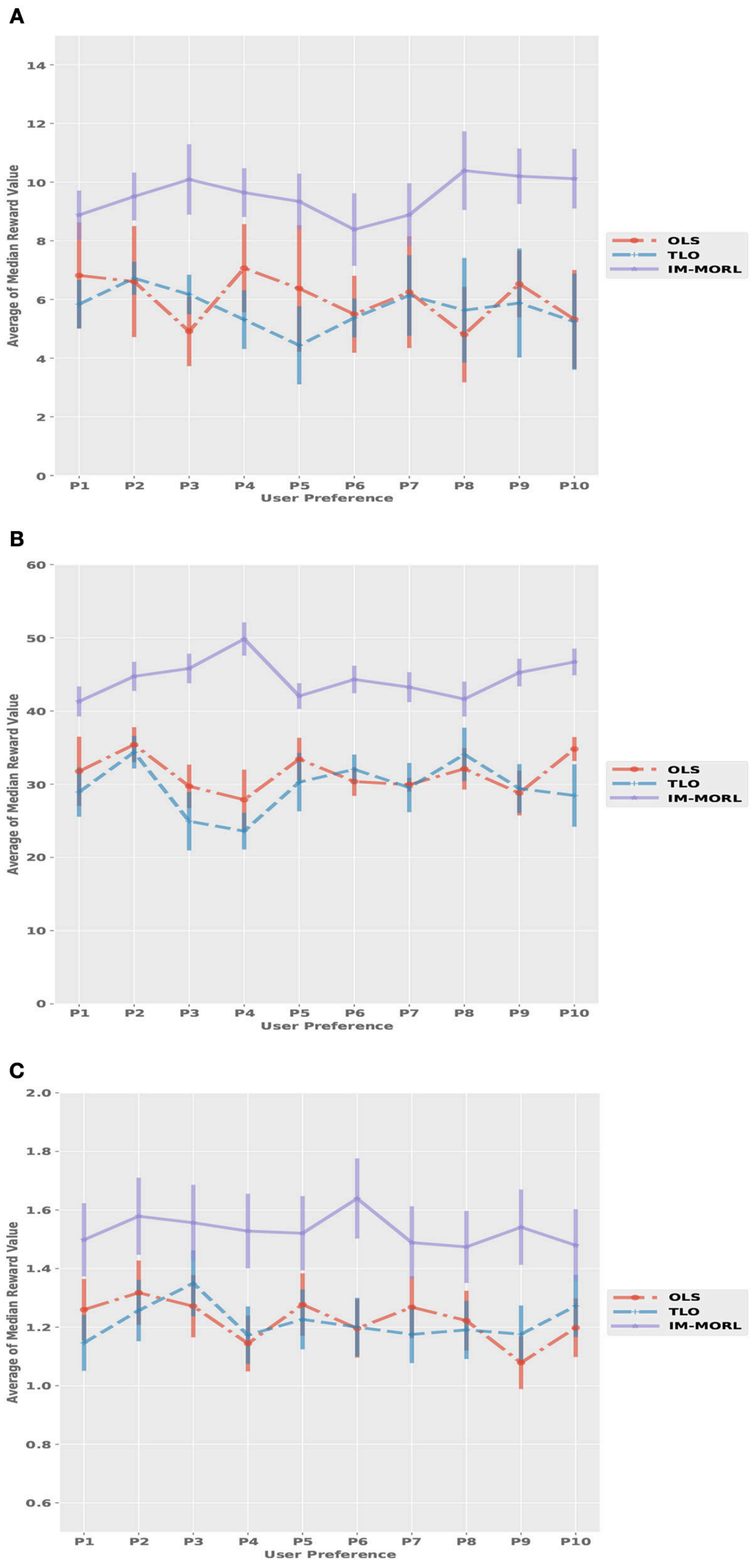


FIGURE 13 | Comparing the median reward values for each user preference averaged over 15 runs with standard deviation bars in the non-stationary environments. **(A)** The search and rescue (SAR) environment. **(B)** The deep sea treasure (DST) environment. **(C)** The resource gathering (RG) environment.

conducted statistical significance t-test between the results of the two agents and it showed that all of them are statistically significant with $p < 0.005$. **Table 4** summarizes these results in terms of average prediction error and standard deviation.

These findings confirms the effectiveness of the proposed intrinsically motivated preference exploration mechanism as it succeeded to sample preferences that can enhance the prediction performance of the predictive model reflecting the stability of CCS policies. While the randomly motivated exploration does not have this ability to guide the search process toward the regions that need enhancements. Basically, it samples preferences uniformly from the weight space without considering the current performance levels of the predictive model or the evolved CCS.

6.2. Comparison to the State-of-the-Art MORL Methods

In this section, we present the results for comparing our IM-MORL agent with agents running two of the state-of-the art MORL methods namely OLS and TLO. As indicated in section 5.1.2, we compare between the three agents using two metrics: the *sum of median rewards* over the ten uniformly sampled user preferences; and the *hypervolume* metric. Firstly, we will present the results for the stationary environments, then for the non-stationary environments afterwards.

6.2.1. Comparison in Stationary Environments

Figure 11 depicts the median reward value for each user preference results averaged over 15 runs with standard deviation bars. While **Table 5** summarizes the average and standard deviation of the sums of median rewards for each run. For the SAR environment shown in **Figure 11A**, the OLS agent achieved an average of 125.3, followed by the TLO agent which achieved an average of 124.7 , finally, our IM-MORL achieved an average of 123.9. Similarly, in the DST environment shown in **Figure 11B**, the OLS agent achieved an average of 539.4, followed by the TLO agent with an average of 536.7, and our IM-MORL agent achieved and average of 535.2. Finally, **Figure 11C** shows the results in the RG environment. The OLS and TLO agents achieved close results of 18.1 and 17.5, respectively, while our IM-MORL agent achieved an average of 16.9.

To asses the statistical significance of the results, we compare the sum of median rewards for each run (15 independent values) over the three comparative methods. We conducted the t-test of statistical significance and found the results are not statistically significant across the three methods ($p > 0.05$).

For the *hypervolume* metric, **Figure 12** presents a bar-chart for comparing results of the three agents grouped by each experimental environment. To neutralize the effect of different reward values per each environment, we show the normalized value of the metric per each environment. In the SAR environment,the OLS agent achieved the highest value of 0.73, followed by the TLO agent with value of 0.67, then our IM-MORL agent with value of 0.65. While in the DST environment, the TLO agent achieved the highest value of 0.85, followed by the OLS agent with value of 0.82, then our IM-MORL agent with value of 0.75. Similarly, in the RG environment, the TLO agent achieved the highest value of 0.58, followed by the OLS agent with value of 0.55, then our IM-MORL agent with value of 0.48. **Table 6** summarizes these results. Similarly, the difference in results was not statistically significant ($p > 0.05$) across the three comparative methods.

6.2.2. Comparison in Non-stationary Environments

For the median reward values, **Figure 13** presents the comparison results between the three agents. While **Table 7** summarizes the average and standard deviation of the sums of median rewards for each run. A common finding in these results is that the IM-MORL agent significantly outperformed the two other agents over the three experimental environment. In the SAR environment, the IM-MORL agent outperformed the OLS and TLO agents by a magnitude of 35.3 and 38.7, respectively. While in the DST environment, the IM-MORL agent outperformed the OLS and TLO agents by a magnitude of 130.6 and 149.2, respectively. Finally, in the RG environment, the IM-MORL agent outperformed the OLS and TLO agents by a average magnitude of 3.1 and 3.5, respectively. All the performance results achieved by IM-MORL agent were statistically significant with $p < 0.05$ in comparison to the two other agents.

The significant performance achieved by the IM-MORL agent was emphasized by the normalized average *hypervolume* results in comparison to the two other agents. **Figure 14** depicts the comparison results for the *hypervolume* metric showing the average and standard deviation over the executed 15 runs and grouped by the experimental environment. In the SAR environment, the IM-MORL agents outperformed the OLS and TLo agents by a average magnitude of 0.29 and 0.24, respectively. While in the DST environment, the IM-MORL agents outperformed the OLS and TLo agents by a average magnitude of 0.34 and 0.39, respectively. Finally in the RG environment, the IM-MORL agents outperformed the OLS and TLo agents by a average magnitude of 0.34 and 0.27, respectively. Conducting the statistical significance test for the results showed that the IM-MORL significantly outperformed the two other agents with $p < 0.05$. **Table 8** summarizes these results.

The finding from results in the non-stationary environments indicate that the IM-MORL agent proved to be more robust and adaptive to non-stationary dynamics in the environment in comparison to the two other state-of-the-art MORL agents. Mainly, there are two main reasons behind this finding.

TABLE 7 | Comparing the OLS, TLO, and IM-MORL agents in terms of sum of median reward values averaged over 15 runs in the non-stationary environments.

Environment	OLS	TLO	IM-MORL
SAR	60.1 ± 5.6	56.7 ± 6.2	**95.4 ± 4.1**
DST	314.2 ± 3.9	295.6 ± 2.8	**444.8 ± 3.4**
RG	12.2 ± 1.2	11.8 ± 1.6	**15.3 ± 1.7**

Bold value indicates best results.

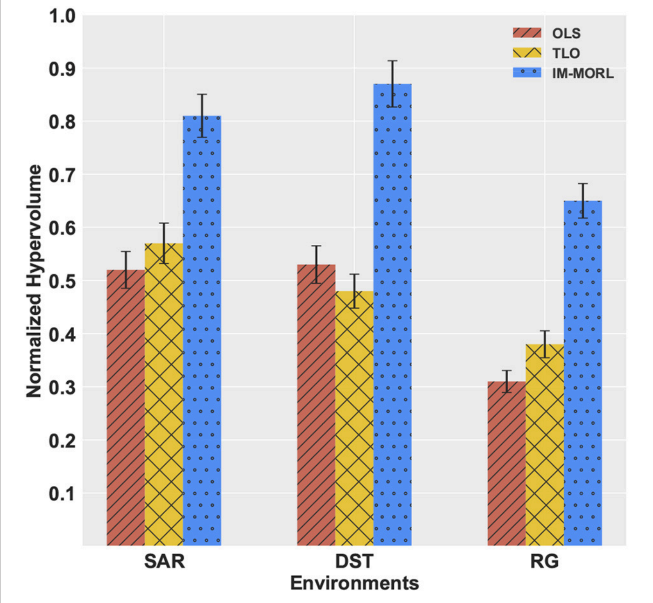


FIGURE 14 | A bar-chart comparing the normalized average *hypervolume* values with standard deviation for the OLS, TLO, and IM-MORL agents grouped by each non-stationary environments.

TABLE 8 | Comparing the OLS, TLO, and IM-MORL agents in terms of average *hypervolume* over 15 runs in the non-stationary environments.

Environment	OLS	TLO	IM-MORL
SAR	0.52 ±0.04	0.57 ±0.03	**0.81 ±0.03**
DST	0.53 ±0.02	0.48 ±0.04	**0.87 ±0.04**
RG	0.31 ±0.03	0.38 ±0.02	**0.65 ±0.02**

Bold value indicates best results.

The first reason is the adaptive preference exploration mechanism of the IM-MORL agent, which is guided by the intrinsic motivation to enhance the performance of the predictive model. This intrinsic motive lead to actively learning the preference areas that the current CCS is not addressing well. During the training phase in the non-stationary environments, this characteristic allowed the IM-MORL agent to re-explore the affected preference regions after changes occur in the environment, while the OLS and TLO agents lack this adaptive preference exploration characteristic. Consequently, they did not adapt sufficiently to the non-stationary dynamics in the environment. An additional note on the performance in the non-stationary is that training on diverse scenarios resulting from changes in the objects location aided the exploration process, which led to evolving better policies during the training phase in comparison to the stationary environments case.

While the second reason is the robustness of the steppingstone policies adopted by the IM-MORL agent to changes in the environment, in comparison to the greedy specialized policies adopted by the OLS and TLO agents. During the non-stationary environments, bootstrapping new policies from steppingstone policies optimized for preference regions adapted better than bootstrapping from policies that were greedily optimized for specific preferences.

7. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel multi-objective reinforcement learning method that is adaptive in non-stationary environments. The proposed method achieves this objective through an adversarial self-play between an intrinsically motivated preference exploration component and a robust policy coverage set optimization component in order to developmentally evolve the optimal convex coverage set that can solve the MOMDP problem. We experimentally assessed the effectiveness of the proposed intrinsically motivated preference exploration and compared our method with two of the state-of-the-art multi-objective reinforcement learning methods over stationary and non-stationary environments. Results showed that there is no statistical significance on the evaluation metrics values in comparison to the two other state-of-the-art methods within the stationary environment, while our proposed method significantly outperformed them in the non-stationary environments.

In the future work of this research, we will investigate how to allow our IM-MORL method to achieve generalization and transfer learning over a varying number of objectives and tasks using hierarchical intrinsically motivated multi-objective policy learning.

AUTHOR CONTRIBUTIONS

SA designed and implemented the algorithms, executed the experiments and visualized the results, and wrote the manuscript. KK and SA designed and the experimental design. JH and KK reviewed the results and provided feedback for presenting the research contribution, provided theoretical guidance for the research. KK reviewed the manuscript and provided feedback.

ACKNOWLEDGMENTS

This research is supported by an Australian Government Research Training Program (RTP) Scholarship.

REFERENCES

Akrour, R., Schoenauer, M., and Sebag, M. (2011). *Preference-Based Policy Learning*. Berlin; Heidelberg: Springer Berlin Heidelberg.

Altman, E. (1999). *Constrained Markov Decision Processes, Vol. 7*. London: CRC Press.

Barto, A. G. (2013). "Intrinsic motivation and reinforcement learning," in *Intrinsically Motivated Learning in Natural and Artificial Systems,* eds G. Baldassarre and M. Mirolli (Berlin; Heidelberg: Springer), 17–47.

Beume, N., Fonseca, C. M., Lopez-Ibanez, M., Paquete, L., and Vahrenhold, J. (2009). On the complexity of computing the hypervolume indicator. *IEEE Trans. Evol. Comput.* 13, 1075–1082. doi: 10.1109/TEVC.2009.2015575

Busa-Fekete, R., Szörényi, B., Weng, P., Cheng, W., and Hüllermeier, E. (2014). Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm. *Mach. Learn.* 97, 327–351. doi: 10.1007/s10994-014-5458-8

Castelletti, A., Pianosi, F., and Restelli, M. (2013). A multiobjective reinforcement learning approach to water resources systems operation: pareto frontier approximation in a single run. *Water Resour. Res.* 49, 3476–3486. doi: 10.1002/wrcr.20295

Deb, K. (2014). *Multi-Objective Optimization*. Boston, MA: Springer.

Feinberg, E. A., and Shwartz, A. (1995). Constrained markov decision models with weighted discounted rewards. *Math. Oper. Res.* 20, 302–320. doi: 10.1287/moor.20.2.302

Fürnkranz, J., Hüllermeier, E., Cheng, W., and Park, S.-H. (2012). Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Mach. Learn.* 89, 123–156. doi: 10.1007/s10994-012-5313-8

Gábor, Z., Kalmár, Z., and Szepesvári, C. (1998). "Multi-criteria reinforcement learning," in *ICML, Vol. 98* (Madison, WI), 197–205.

Geibel, P. (2006). "Reinforcement learning for MDPs with constraints," in *ECML, Vol. 4212* (Heidelberg: Springer), 646–653.

Kriegel, H.-P., Kröger, P., Sander, J., and Zimek, A. (2011). Density-based clustering. *Wiley Interdisc. Rev.* 1, 231–240. doi: 10.1002/widm.30

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521:436. doi: 10.1038/nature14539

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2016). "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)* (San Juan).

Lizotte, D. J., Bowling, M. H., and Murphy, S. A. (2010). "Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (Citeseer), 695–702.

Merrick, K. E., and Maher, M. L. (2009). *Motivated Reinforcement Learning: Curious Characters for Multiuser Games*. Berlin: Springer Science & Business Media.

Ming, L., and Vitányi, P. (1997). *An Introduction to Kolmogorov Complexity and Its Applications*. Berlin: Springer Heidelberg.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518:529. doi: 10.1038/nature14236

Moffaert, K. V., Drugan, M. M., and Nowé, A. (2013). "Scalarized multi-objective reinforcement learning: novel design techniques," in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)* (Singapore), 191–199.

Mossalam, H., Assael, Y. M., Roijers, D. M., and Whiteson, S. (2016). Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*.

Ogryczak, W., Perny, P., and Weng, P. (2011). "On minimizing ordered weighted regrets in multiobjective markov decision processes," in *Algorithmic Decision Theory*, eds R. I. Brafman, F. S. Roberts, and A. Tsoukiàs (Berlin; Heidelberg:Springer Berlin Heidelberg), 190–204.

Oudeyer, P.-Y., and Kaplan, F. (2009). What is intrinsic motivation? a typology of computational approaches. *Front. Neurorobot.* 1:6. doi: 10.3389/neuro.12.006.2007

Oudeyer, P. Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evol. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Papadimitriou, C. H., and Tsitsiklis, J. N. (1987). The complexity of markov decision processes. *Math. Oper. Res.* 12, 441–450. doi: 10.1287/moor.12.3.441

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning (ICML)* (Sydney).

Perny, P., and Weng, P. (2010). "On finding compromise solutions in multiobjective markov decision processes," in *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence* (Amsterdam: IOS Press), 969–970.

Qureshi, A. H., Nakamura, Y., Yoshikawa, Y., and Ishiguro, H. (2018). Intrinsically motivated reinforcement learning for human-robot interaction in the real-world. *Neural Netw.* doi: 10.1016/j.neunet.2018.03.014. [Epub ahead of print].

Roijers, D. M., Vamplew, P., Whiteson, S., and Dazeley, R. (2013). A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.* 48, 67–113. doi: 10.1613/jair.3987

Roijers, D. M., and Whiteson, S. (2017). A survey of multi-objective sequential decision-making. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 34.

Roijers, D. M., Whiteson, S., and Oliehoek, F. A. (2014). "Linear support for multi-objective coordination graphs," in *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems* (Richland, SC), 1297–1304.

Roijers, D. M., Whiteson, S., and Oliehoek, F. A. (2015). "Point-based planning for multi-objective pomdps," in *IJCAI* (Buenos Aires), 1666–1672.

Ryan, R. M., and Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemp. Educ. Psychol.* 25, 54–67. doi: 10.1006/ceps.1999.1020

Schmidhuber, J. (1991a). "Curious model-building control systems," in *IEEE International Joint Conference on Neural Networks, 1991* (Seattle, WA), 1458–1463.

Schmidhuber, J. (1991b). "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats* (Paris), 222–227.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Trans. Auton. Mental Dev.* 2, 230–247. doi: 10.1109/TAMD.2010.2056368

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature* 529, 484–489. doi: 10.1038/nature16961

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of go without human knowledge. *Nature* 550, 354–359. doi: 10.1038/nature24270

Singh, S., Lewis, R. L., and Barto, A. G. (2009). "Where do rewards come from," in *Proceedings of the Annual Conference of the Cognitive Science Society* (Amsterdam), 2601–2606.

Sutton, R. S., and Barto, A. G. (1998). *Introduction to Reinforcement Learning, Vol. 135*. Cambridge: MIT Press.

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., and Dekker, E. (2011). Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Mach. Learn.* 84, 51–80. doi: 10.1007/s10994-010-5232-5

Watkins, C. J. C. H., and Dayan, P. (1992). Q-learning. *Mach. Learn.* 8, 279–292.

Zadeh, L. A. (1996). Fuzzy logic = computing with words. *IEEE Trans. Fuzzy Syst.* 4, 103–111. doi: 10.1109/91.493904

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH
published: 16 October 2018
doi: 10.3389/fnbot.2018.00066

Robot End Effector Tracking Using Predictive Multisensory Integration

Lakshitha P. Wijesinghe[1]*, *Jochen Triesch*[2] and *Bertram E. Shi*[1]

[1] *Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong,* [2] *Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany*

We propose a biologically inspired model that enables a humanoid robot to learn how to track its end effector by integrating visual and proprioceptive cues as it interacts with the environment. A key novel feature of this model is the incorporation of sensorimotor prediction, where the robot predicts the sensory consequences of its current body motion as measured by proprioceptive feedback. The robot develops the ability to perform smooth pursuit-like eye movements to track its hand, both in the presence and absence of visual input, and to track exteroceptive visual motions. Our framework makes a number of advances over past work. First, our model does not require a fiducial marker to indicate the robot hand explicitly. Second, it does not require the forward kinematics of the robot arm to be known. Third, it does not depend upon pre-defined visual feature descriptors. These are learned during interaction with the environment. We demonstrate that the use of prediction in multisensory integration enables the agent to incorporate the information from proprioceptive and visual cues better. The proposed model has properties that are qualitatively similar to the characteristics of human eye-hand coordination.

Keywords: active efficient coding, developmental robotics, sensorimotor prediction, generative adaptive subspace self-organizing map, reinforcement learning

OPEN ACCESS

Edited by:
Pierre-Yves Oudeyer, Institut National de Recherche en Informatique et en Automatique (INRIA), France

Reviewed by:
Matej Hoffmann, Faculty of Electrical Engineering, Czech Technical University in Prague, Czechia
Kazutaka Takahashi, University of Chicago, United States
Bruno Lara, Universidad Autónoma del Estado de Morelos, Mexico

****Correspondence:***
Lakshitha P. Wijesinghe
lpwijesinghe@ust.hk

Received: *31 May 2018*
Accepted: *20 September 2018*
Published: *16 October 2018*

Citation:
Wijesinghe LP, Triesch J and Shi BE (2018) Robot End Effector Tracking Using Predictive Multisensory Integration. Front. Neurorobot. 12:66. doi: 10.3389/fnbot.2018.00066

INTRODUCTION

To perform complex manipulation tasks, conventional robotic systems require precise calibration, which must be repeated when their physical configuration changes. In contrast, humans learn manipulation skills autonomously, and automatically recalibrate in response to physical configuration changes, e.g., due to growth and injury. Eye-hand coordination is a key skill required for these tasks. It requires the integration of multiple sensory modalities, such as vision and proprioception. Human infants appear to learn to develop a sense of themselves through observing the temporal contingency and spatial congruency of the sensory (e.g., visual, auditory, and proprioceptive) feedback received during self-produced motion, such as motor babbling (Rochat, 1998). One goal of cognitive developmental robots is to endow robots with this capability so that they will not require any manual calibration before acting in a new environment (Asada et al., 2009).

The mismatch between the motion of objects in the environment and the eye's rotational velocity creates retinal slip. During tracking motions, such as smooth pursuit, the brain attempts to minimize this slip by adapting the eye rotational velocity. Motion in the environment is generated by either self-motion (e.g., of the hand) or exteroceptive motion. When the hand moves, its motion can be sensed via two sensory modalities: retinal slip caused by the relative motion between the hand and eye and proprioceptive sensing of the position and movement of the arm. In contrast, an external object moving in the environment only generates a retinal slip. Moreover, hand motion in total darkness only provides proprioceptive information to the brain. In such conditions, the human brain has the ability to generate eye movements to follow the hand or an external target using smooth pursuit like eye movements.

In this paper, we propose a novel predictive model for learning robotic visuomotor control. The proposed system model is inspired by recent findings that neurons in the primary visual cortex (area V1) are driven not only by visual but also by the motor input. Activity in V1 was long believed to be driven only by visual inputs. However, recent findings on visual perception in awake mice have suggested that this is not true. For example, the responses in V1 depend on behavioral state (Niell and Stryker, 2011). Experiments conducted in darkness revealed that motor activity alone could trigger responses in V1 neurons (Saleem et al., 2013). The development of V1 depends upon visuomotor coupling (Attinger et al., 2017). Most relevant to this work is the discovery of cells that respond to the mismatch between the actual and predicted visual flow (Keller et al., 2012; Zmarz and Keller, 2016). This suggests that visual areas predict the sensory consequences of motor actions.

Our model takes in visual input from a camera and proprioceptive inputs from the encoders of the robot arm, and produces eye motor actions to track the moving robot hand. The model is based upon the hypothesis that the brain utilizes proprioceptive inputs to predict the visual consequences of motor actions. In line with other work in predictive coding, we use the term "predict" to refer to the process of generating an estimate of one sensory input from the values of other inputs, which may occur at the same time, rather than a more strict definition where future values are estimated from past and present values. The prediction is often used to generate a mismatch signal by comparison with the actual input. For example, Srinivasan et al. (1982) explain center-surround antagonism in the retina using predictive coding, where the predicted intensity at the center based on the surround is subtracted from the actual center signal. Rao and Ballard (1999) predict lower level cortical outputs from higher level cortical outputs. Zmarz and Keller (2016) find mismatch neurons that respond to the difference between the actual visual flow and the prediction of visual flow from self-motion. Our model is most similar to the latter work, where prediction is across sensory modalities.

There are several important novel attributes of the learning methodology compared to similar work in the literature. First, the learning does not depend on any fiducial visual marker to identify the end effector of the robot. Second, the model does not require the forward kinematics of the arm to be known. Third, pre-defined visual feature descriptors are not required, but rather are learned. Finally, our experimental results with this model suggest that the use of prediction enables the model to better integrate proprioception and vision.

The rest of the paper is organized as follows. In section Related Work, we place our work into the context of past work. Section Materials and Methods describes the model, experimental setup and learning algorithms. Then in section Results, we present experimental results comparing the tracking performance. We also compare our model characteristics with human psychophysical data. Finally, in section Discussion, we further discuss the results presenting the corresponding conclusions.

RELATED WORK

The problem of learning end effector tracking is a part of the larger problem of autonomous learning of the body schema. The body schema is a sensorimotor representation of the body that can be used to direct motion and actions. It integrates multiple cues, including proprioception, vision, audition, vestibular cues, tactile cues, and motor cues, to represent the relations between the spatial positions of the body parts. Knowledge of the body schema can be used in a number of different tasks, e.g., end effector tracking, reaching, posture control and locomotion.

The review by Hoffmann et al. (2010) classifies body schema representations used in robotics into two classes: explicit and implicit. Both have been used to address the problem of end effector tracking. In the explicit approach (e.g., Bennett et al., 1991; Hollerbach and Wampler, 1996; Gatla et al., 2007), transformations between sensory and motor coordinates are broken down into a chain of closed form transformations where each link corresponds explicitly to part of the robot structure. The work we present here falls into the class of implicit models, where an implicit representation (e.g., a look up table or neural network) is used.

Past work has often used a point representation of the end effector, where artificial markers (e.g., color blobs) have been used to enable easy identification of the end effector (Hersch et al., 2008; Sturm et al., 2009). For example, a biologically inspired model to learn visuomotor coordination for the robot Nao was proposed in Schillaci et al. (2014). Learning occurred during motor babbling, which is similar to how infants may learn early eye-hand coordination skills. The proposed method used two Dynamic Self Organizing Maps (DSOMs) to represent the arm and neck position of the robot. The connections between the DSOMs were strengthened if the robot was looking at a fiducial marker positioned on the end effector. After learning, the robot had the ability to track the end effector by controlling the neck joints. One advantage of this model is that the method has no assumption that the forward arm kinematics of the robot is known. However, one limitation of the approach is that it required a fiducial marker.

Subsequent work has relaxed the assumption that the end effector is a point and removed the requirements for explicit markers. However, it has still required hard-coded visual feature descriptors. For example, an algorithm to learn the mapping from arm joint space to the corresponding region in image space containing the end effector was proposed in Zhou and Shi (2016), based on a measure of visual consistency defined using SIFT features (Lowe, 2004). This algorithm did not require prior knowledge of the arm model, and was robust to changes in the appearance of the end effector. Other marker-less approaches have relied upon knowledge of a 3D CAD model of the end effector (Vicente et al., 2016; Fantacci et al., 2017). Vicente et al. (2016) eliminated calibration errors using a particle filter. The likelihood associated with each particle was evaluated by comparing the outputs of Canny edge detectors applied to both the real and simulated camera images. Fantacci et al. (2017) extended this particle filter and 3D CAD model based approach to estimate the end effector pose. The likelihood was evaluated using a Histogram of Oriented Gradient (HOG) (Dalal and Triggs, 2005) based transformation to compare the two images. The approach to bootstrap a kinematic model of a robot arm proposed in Broun et al. (2014) does not require a priori knowledge of a CAD model, as it constructs a model of the end-effector on the fly from Kinect point cloud data. However, it still requires a hard-coded optical flow extraction stage to identify the arm in the image through visuomotor correlation.

Some of the limitations in the aforementioned research (e.g., the requirement for a marker and/or hard-coded image features) were addressed in our prior work (Wijesinghe et al., 2017), which proposed a multisensory neural network that combined visual and proprioceptive modalities to track a robot arm. Retinal slip during the motion was represented by encoding two temporally consecutive image frames using a sparse coding algorithm where the basis vectors were learned online (Zhang et al., 2014). The sparse coefficients were combined with proprioceptive input to control the eye to track the arm. This paper extends our previous idea by introducing a new model following the hypothesis that the brain generates internal predictions for consequences of actions.

MATERIALS AND METHODS

Our approach is based on the Active Efficient Coding (AEC) framework (Zhao et al., 2012; Teulière et al., 2015), a generalization of the efficient coding hypothesis to active perception. Under the efficient coding hypothesis, the sensory data is encoded efficiently by exploiting redundancies in the statistics of the sensory input signals. In AEC, movements of the sensory organs are also learned so that the inputs can be coded efficiently. In the proposed model, visual, and proprioceptive stimuli are jointly encoded. This perceptual representation is used to generate eye movements for tracking the robot arm. Simulation of the model is performed using the iCub humanoid robot simulator, an open source robot simulator for the iCub robot (Tikhanoff et al., 2008). We provide more detail in the following subsections.

Model Architecture

Figure 1 illustrates the architecture of the proposed model, which evolves in discrete time. We assume that each iteration corresponds to 40 ms.

At each iteration, the right eye of the iCub captures an image with 320 × 240 pixel resolution. Two foveal subwindows are extracted from the center of this image: a smaller 55 × 55 pixel fine scale image, $I_{\text{fine}}(t)$, and a larger 110 × 110 pixels coarse scale subwindow, $I_{\text{coarse}}(t)$, which is subsampled horizontally and vertically by a factor of two. These subwindows cover 11^o and 22^o, respectively, horizontally and vertically.

The visual stimuli are encoded using Generative Adaptive Subspace Self Organizing Maps (GASSOMs) (Chandrapala and Shi, 2015). Proprioceptive inputs include the arm position, velocity, and acceleration ($\boldsymbol{\theta}_a(t) \in \mathbb{R}^4, \dot{\boldsymbol{\theta}}_a(t) \in \mathbb{R}^4, \ddot{\boldsymbol{\theta}}_a(t) \in \mathbb{R}^4$) and the eye position and eye velocity ($\boldsymbol{\theta}_e(t) \in \mathbb{R}^2, \dot{\boldsymbol{\theta}}_e(t) \in \mathbb{R}^2$) as reported by the motor encoders. The prediction module predicts the sensory consequences of the arm and eye motions based on proprioceptive inputs. This enables the model to reduce the correlation between the visual features and the proprioceptive inputs during self-motion. Finally, the visual and proprioceptive inputs are integrated using an Artificial Neural Network (ANN) to generate pan and tilt eye acceleration commands, $\hat{\ddot{\boldsymbol{\theta}}}_e(t) \in \mathbb{R}^2$, enabling the right eye to track the robot arm. The "∧" symbol is added to distinguish motor commands from proprioceptive information.

Given the eye acceleration command, the eye velocity is obtained by;

$$\hat{\dot{\boldsymbol{\theta}}}_e(t+1) = \hat{\dot{\boldsymbol{\theta}}}_e(t) + \hat{\ddot{\boldsymbol{\theta}}}_e(t). \tag{1}$$

Equation (1) is similar to the model for the maintenance of pursuit described in Lisberger (2010), where an efference copy of the eye velocity command is fed back in order to determine the current command for eye velocity in the immediate future. This enables eye velocity to be maintained automatically. In our model, both image motion and arm motion can drive changes in eye velocity through the eye acceleration command.

The model presented here uses only afferent information to determine the eye acceleration. In biological systems, both afferent and efferent signals from the arm are involved in arm-eye coordination control. For deafferented monkeys, smooth pursuit eye movements disappeared while tracking a target moved by active arm movements in darkness (Gauthier and Mussa Ivaldi, 1988). Steinbach (1969) found differences in ocular tracking of active and passive hand motions, which suggest that efference commands also play a crucial role. Gauthier and Mussa Ivaldi (1988) and Gauthier et al. (1988) suggested that efferent signals serve to synchronize the onsets of arm and eye motions, whereas proprioceptive signals serve to couple the eye and hand motor signals once movement has started. Subsequent experiments have provided additional support for this hypothesis (Vercher et al., 1996). Since we use only afferent information, the model may provide an account for differences in performance once movement has started. We leave the integration of an efference copy to future extensions of the model.

During training, we control four degrees of freedom (DoF) among the seven DoF in the iCub robot arm. The three joints in the wrist are fixed, and the remaining four joints (the shoulder pitch, shoulder roll, shoulder yaw, and the elbow joint) are controlled. We fix the wrist angles so that the palm of the iCub robot remains approximately parallel to the image plane. As described below, one assumption of our approach is that the retinal flow is uniform across both fine and coarse foveal image regions. Keeping the wrist angle fixed ensures that the image of the palm covers the foveal images when the gaze vector intersects the center of the palm. Modifying the algorithm so that the image region used to generate motor commands vary in size automatically may enable the algorithm to allow all DoF to vary. During testing, we allow the wrist to move (see section Qualitative Evaluation of Performance).

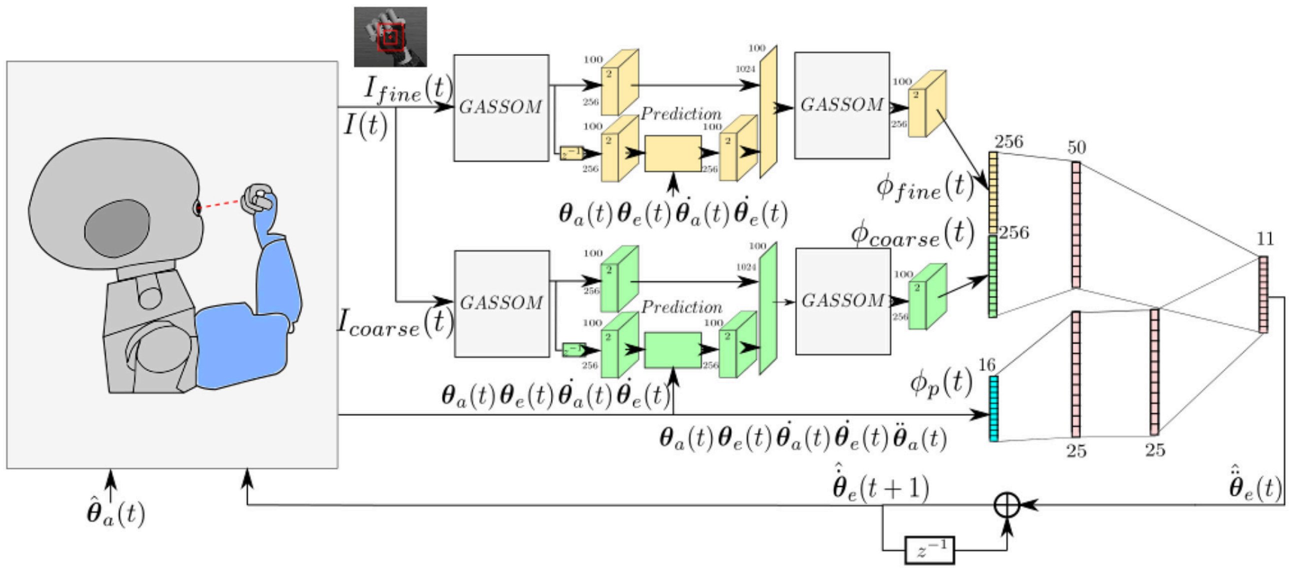

$I_{fine}(t)$
$I(t)$
$I_{coarse}(t)$
GASSOM
Prediction
$\boldsymbol{\theta}_a(t)\boldsymbol{\theta}_e(t)\dot{\boldsymbol{\theta}}_a(t)\dot{\boldsymbol{\theta}}_e(t)$
$\phi_{fine}(t)$
$\phi_{coarse}(t)$
$\phi_p(t)$
$\boldsymbol{\theta}_a(t)\boldsymbol{\theta}_e(t)\dot{\boldsymbol{\theta}}_a(t)\dot{\boldsymbol{\theta}}_e(t)\ddot{\boldsymbol{\theta}}_a(t)$
$\hat{\boldsymbol{\theta}}_a(t)$
$\hat{\dot{\boldsymbol{\theta}}}_e(t+1)$
$\hat{\ddot{\boldsymbol{\theta}}}_e(t)$
z^{-1}
256
50
256
11
16
25
25

FIGURE 1 | The model takes as input images at two scales and proprioceptive readings from the encoders. The eye controller represented by the neural network maps the perceptual representation to motor actions.

In the following, we describe the model components in more detail.

Visual and Proprioceptive Features

The model encodes the visual stimuli in two stages. In the first stage, each foveal image is divided into a 10×10 array of 10×10 pixel overlapping patches, $\mathbf{x}_{1,s,i}(t) \in \mathbb{R}^{100}$, with a stride of 5 pixels, where s indexes scale and $i \in \{1, 2, \ldots, P\}$ indexes the patch ($P = 100$). The subscript "1" indicates the stage.

Patches are encoded using GASSOMs (Chandrapala and Shi, 2015). The GASSOM is a probabilistic generative extension of the Adaptive Subspace Self Organizing Map (ASSOM) (Kohonen, 1996). It assumes that each input vector $\mathbf{x}_{1,s,i}(t)$ is generated by one of $N = 256$ nodes. The generating nodes evolve according to a first order Markov process. Each node is associated with a two dimensional subspace spanned by basis vectors specified by the orthogonal columns of the matrix $\mathbf{B}_{1,s,n} \in \mathbb{R}^{100\times2}$. The input is generated by the node by choosing a Gaussian distributed vector lying on the subspace plus a small noise vector lying in the orthogonal complement of the subspace. Using the algorithm described in Chandrapala and Shi (2015), both the transition matrix of the Markov process and the node subspaces are learned so as to maximize the likelihood of the input sensory data. The learned transition matrices have high self-transition probability, which implies that the node generating input $t - 1$ is likely to generate the input at time t, a property we refer to as "slowness."

The output of the GASSOM is the set of projections of the input vector onto the N subspaces.

$$\mathbf{p}_{1,s,n,i}(t)=\mathbf{B}^{T}_{1,s,n}\,\mathbf{x}_{1,s,i}(t). \tag{2}$$

When the eye is viewing the end effector, these projections change in a regular manner, which depends upon the movement of the arm and the eye.

Each node at each scale has an associated prediction module, which predicts the projection of the input at time t, $\mathbf{p}_{1,s,n,i}(t)$, given the input at time $t - 1$, $\mathbf{p}_{1,s,n,i}(t - 1)$ and the proprioceptive signals encoding the arm/eye position/velocity $(\boldsymbol{\theta}_a(t), \dot{\boldsymbol{\theta}}_a(t), \boldsymbol{\theta}_e(t), \dot{\boldsymbol{\theta}}_e(t))$. **Figure 2** indicates the projections of the patch $\mathbf{x}_{1,s,i}$ at times t and $t - 1$ and the corresponding transformation in the subspace. The prediction module assumes the transformation can be modeled as a linear mapping, where the predicted projection at time t is given by;

$$\hat{\mathbf{p}}_{1,s,n,i}(t) = \begin{bmatrix} \alpha_{s,n} & \beta_{s,n} \\ \gamma_{s,n} & \delta_{s,n} \end{bmatrix} \mathbf{p}_{1,s,n,i}(t-1), \tag{3}$$

where the α, β, γ and δ parameters for each scale s and node n depend upon the arm/eye position/velocity. These parameters are computed using a neural network with four inputs $(\boldsymbol{\theta}_a(t), \dot{\boldsymbol{\theta}}_a(t), \boldsymbol{\theta}_e(t), \dot{\boldsymbol{\theta}}_e(t))$, one hidden layer containing 25 hidden units with tanh activations, and four linear output neurons. Since all patches share the same α, β, γ, and δ parameters, we are assuming that the retinal flow is uniform across the foveal images. By performing the prediction in the projected subspace, rather than the original high dimensional pixel space, we simplify the task of prediction.

The parameters of the 512 (2 scales $\times$ 256 nodes) neural networks are learned online using stochastic gradient descent, where the weights are updated every iteration. Since each foveal image contains 100 patches, we average the gradients of the prediction error across the 100 patches, and update the weights of each neural network with the average gradient.

The second GASSOM encodes the concatenated vectors $\mathbf{p}_{1,s,n,i}(t)$ and $\hat{\mathbf{p}}_{1,s,n,i}(t)$ corresponding to all the nodes $n \in \{1, 2, \ldots, 256\}$ in the first GASSOM. Hence, the input vector to the second GASSOM is $\mathbf{x}_{2,s,i}(t) \in \mathbb{R}^{1024}$ for a given scale

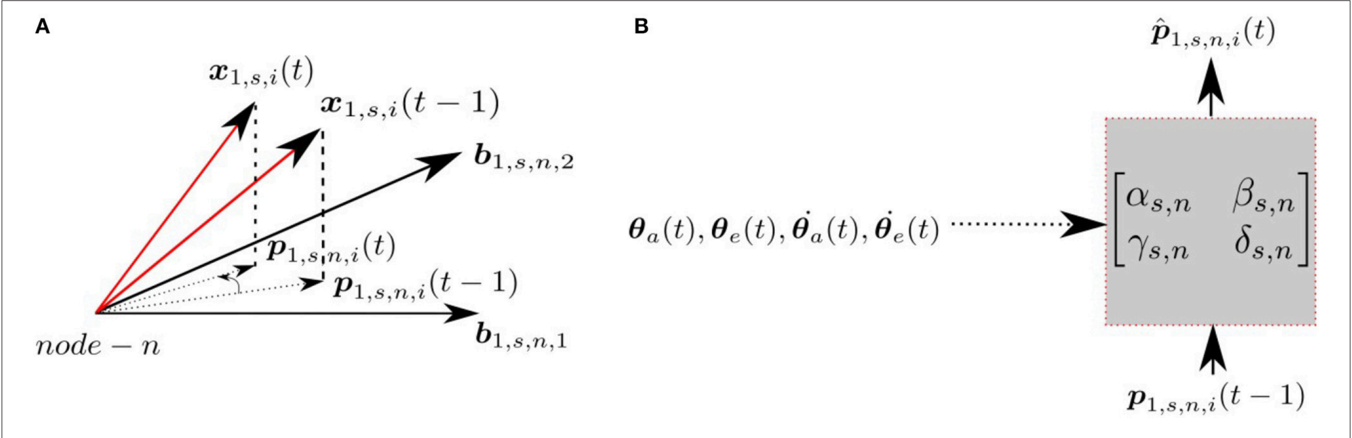

A
B
$\boldsymbol{x}_{1,s,i}(t)$
$\boldsymbol{x}_{1,s,i}(t-1)$
$\boldsymbol{b}_{1,s,n,2}$
$\boldsymbol{p}_{1,s,n,i}(t)$
$\boldsymbol{p}_{1,s,n,i}(t-1)$
$\boldsymbol{b}_{1,s,n,1}$
$node-n$
$\hat{\boldsymbol{p}}_{1,s,n,i}(t)$
$\boldsymbol{\theta}_a(t), \boldsymbol{\theta}_e(t), \dot{\boldsymbol{\theta}}_a(t), \dot{\boldsymbol{\theta}}_e(t)$
$\begin{bmatrix}\alpha_{s,n} & \beta_{s,n}\\ \gamma_{s,n} & \delta_{s,n}\end{bmatrix}$
$\boldsymbol{p}_{1,s,n,i}(t-1)$

FIGURE 2 | (A) Projections at time **t** and **t** − **1** for a given node **n**, scale **s** and patch **i**. **(B)** The transformation matrix projecting current projection at time **t**, given the projection at time **t** − **1** with the proprioceptive input.

s and patch i. The second GASSOM also contains $M = 256$ nodes, each with an associated 2-dimensional subspace spanned by the columns of the matrix $\mathbf{B}_{2,s,m} \in \mathbb{R}^{1024\times 2}$ where $m \in \{1, 2, \dots, 256\}$ indexes the node. As in Equation (2), the projections are computed by;

$$\mathbf{p}_{2,s,m,i}(t) = \mathbf{B}_{2,s,m}^T \mathbf{x}_{2,s,i}(t). \tag{4}$$

Both the transition matrix and the node subspaces are learned online as the iCub behaves in the environment.

From the projections $\mathbf{p}_{2,s,m,i}(t) \in \mathbb{R}^2$ at each node, we compute a feature vector $\boldsymbol{\phi}_s(t) \in \mathbb{R}^{256}$ for each scale of the foveal image by computing the average squared length of the projections over all the patches.

$$\boldsymbol{\phi}_s(t) = \begin{bmatrix} \frac{1}{P}\sum_{i=1}^{P} \left\| \mathbf{p}_{2,s,1,i}(t) \right\|^2 \\ \frac{1}{P}\sum_{i=1}^{P} \left\| \mathbf{p}_{2,s,2,i}(t) \right\|^2 \\ \cdots \\ \frac{1}{P}\sum_{i=1}^{P} \left\| \mathbf{p}_{2,s,256,i}(t) \right\|^2 \end{bmatrix}. \tag{5}$$

The final feature representation of the visual stimuli is the concatenation of the feature vectors at the two scales.

$$\boldsymbol{\phi}_v(t) = \begin{bmatrix} \boldsymbol{\phi}_{\text{fine}}(t) \\ \boldsymbol{\phi}_{\text{coarse}}(t) \end{bmatrix}. \tag{6}$$

The proprioceptive feature vector $\boldsymbol{\phi}_p(t) \in \mathbb{R}^{16}$ concatenates the arm position, velocity and acceleration measurements from the encoders, $\boldsymbol{\theta}_a(t), \dot{\boldsymbol{\theta}}_a(t), \ddot{\boldsymbol{\theta}}_a(t) \in \mathbb{R}^4$ and the eye position and velocity $\boldsymbol{\theta}_e(t), \dot{\boldsymbol{\theta}}_e(t) \in \mathbb{R}^2$. Each proprioceptive input is normalized by subtracting the mean and dividing by the standard deviation computed over the training data set.

Eye Motor Controller

The eye controller maps the visual $\boldsymbol{\phi}_v(t) \in \mathbb{R}^{512}$ and proprioceptive $\boldsymbol{\phi}_p(t) \in \mathbb{R}^{16}$ feature vectors to an eye acceleration command $\ddot{\hat{\boldsymbol{\theta}}}_e(t)$ using the artificial neural network shown in **Figure 1**. Only one network is shown, corresponding to the generation of the acceleration command for one axis (pan or tilt). The other axis is controlled by a network with the same structure, but different weights.

Each neural network has 11 output neurons, corresponding to 11 possible acceleration actions, $a_i \in \mathbf{A}$ for $i \in \{0, 1, \dots, 11\}$, where;

$$\mathbf{A} = \{-1.6, -0.8, -0.4, -0.2, -0.1, 0, 0.1, 0.2, 0.4, 0.8, 1.6\}\ deg/sample^2. \tag{7}$$

The outputs of each neural network, $\pi_{k,i}(t)$ where $k \in \{\text{pan}, \text{tilt}\}$ and $i \in \{0, 1, \dots, 11\}$ encode the probabilities that the actions are chosen at each time t. Mathematically;

$$P\left[\ddot{\hat{\theta}}_{e,k}(t) = a_i\right] = \pi_{k,i}(t), \tag{8}$$

where $\ddot{\hat{\boldsymbol{\theta}}}_e(t) = \left[\ddot{\hat{\theta}}_{e,\text{pan}}(t)\ \ddot{\hat{\theta}}_{e,\text{tilt}}(t)\right]^T$. The eye acceleration command is generated by sampling from this probability distribution.

Within the neural network, the visual input first passes through a single fully connected 50 neuron hidden layer and the proprioceptive input first passes through two fully connected 25 neuron hidden layers with tanh activations before the two pathways are combined at the output layer, which is fully connected with a softmax output non-linearity. Mathematically;

$$\pi_{k,i}(t) = \frac{\exp(z_{k,i}(t)/\tau)}{\sum_{j=1}^{11} \exp(z_{k,j}(t)/\tau)}, \tag{9}$$

where $\tau = 1$ is a temperature parameter. The vector, $\mathbf{z}_k(t) \in \mathbb{R}^{11}$, is given by;

$$\mathbf{z}_k(t) = \mathbf{W}_{k,2}^T \tanh\left(\mathbf{W}_{k,1}^T \boldsymbol{\phi}_v(t)\right) + \mathbf{W}_{k,5}^T \tanh\left(\mathbf{W}_{k,4}^T \tanh\left(\mathbf{W}_{k,3}^T \boldsymbol{\phi}_p(t)\right)\right), \tag{10}$$

where $\mathbf{W}_{k,1} \in \mathbb{R}^{512\times 50}, \mathbf{W}_{k,2} \in \mathbb{R}^{50\times 11}, \mathbf{W}_{k,3} \in \mathbb{R}^{16\times 25}, \mathbf{W}_{k,4} \in \mathbb{R}^{25\times 25}$ and $\mathbf{W}_{k,5} \in \mathbb{R}^{25\times 11}$ are weight matrices. Our implementation includes constant bias terms at all layers, which we have not shown explicitly in the notation to avoid clutter.

The weights of the eye motor controller are learned online as the iCub behaves in the environment, using the natural actor-critic reinforcement learning algorithm (Bhatnagar et al., 2009). The network generating the probabilities described above is the actor (policy) network. The actor-critic algorithm also requires a second network to approximate the value function, which depends upon both $\boldsymbol{\phi}_{\mathrm{v}}(t)$ and $\boldsymbol{\phi}_{\mathrm{p}}(t)$. We use a single layer linear network with 528 inputs feeding into a single linear output neuron for the critic network.

The instantaneous reward is given by;

$$r(t) = -\frac{1}{2}\left(e_{\text{fine}}(t) + e_{\text{coarse}}(t)\right), \tag{11}$$

where;

$$e_s(t) = \frac{1}{P}\sum_{i=1}^{P} \max_{n} \left\| \mathbf{p}_{1,s,n,i}(t) - \mathbf{p}_{1,s,n,i}(t-1)\right\|^{\mathbf{2}}, \tag{12}$$

for $s \in \{\text{fine}, \text{coarse}\}$. This reward penalizes changes in the patch projections, which should be constant if the image of the end effector is stabilized.

Training and Testing Environment

Training runs for a total of 400,000 iterations. It interleaves two different types of sessions (**Figure 3**), each lasting 5,000 iterations at a time. For both session types, the environment contains a planar object mapped with a natural image texture chosen at random from the (Olmos and Kingdom, 2004) database of size 2142 × 1422 pixels located at 0.4–0.8 m distance in front of the iCub. When placed directly in front of the robot, the plane subtends 136° of visual angle horizontally and 118° of visual angle vertically. The texture is changed every 500 iterations. In session type 1, the arm remains stationary in a position where it is outside the field of view of the robot. Only the textured plane, which moves randomly horizontally and vertically, is visible. In session type 2, the robot arm babbles so that the end effector moves randomly in front of the iCub. Depending on the eye and arm position, the center of the eye gaze may fall on the robot arm or on the resting plane. If the eye gaze falls on the center of the palm of the iCub, the hand fills both the coarse and fine scale foveal windows, but this is not always the case. This training setup is intended to mimic a general environment, where an agent is exposed to both self-generated and exteroceptive motion in the visual environment.

In session type 2, we use motor babbling to generate the visual and proprioceptive data for the learning algorithm. The arm babbles around a home pose $\boldsymbol{\theta}_{\mathrm{a}}^{\mathrm{h}} = [-82^{o}\ 22^{o}\ 40^{o}\ 90^{o}]$, which is chosen so that the center of the iCub's hand falls on the image center of the right eye when its pan and tilt angles are zero. The arm moves through a randomly generated trajectory ($\hat{\boldsymbol{\theta}}_{\mathrm{a}}(t)$ in **Figure 1**) in the arm joint space. The babbling trajectory is generated by feeding a set of via points sampled from a uniform distribution $[\boldsymbol{\theta}_{\mathrm{a,i}}^{\mathrm{h}} - 12^{o}, \boldsymbol{\theta}_{\mathrm{a,i}}^{\mathrm{h}} + 12^{o}]$ for $i \in \{1,2,3,4\}$ into the "mstraj" function of the Robotics Toolbox (Corke, 2017) to generate a trajectory consisting of linear segments connected by polynomial blends.

In session type 1, the planar object follows a trajectory created by first generating an arm trajectory as described above, and then moving the plane so that its center point follows the same angular trajectory as the center of the iCub's hand. This ensures that the statistics of the visual motion induced by the plane are similar to the statistics of those induced by the hand.

The eye rotational angles are restricted to $\pm 40^{o}$ and $\pm 30^{o}$ in pan and tilt, respectively. The rotational velocity of the eye is also limited to ± 3 deg/sample in both pan and tilt. During training, we reset the eye position to a "home" position $\boldsymbol{\theta}_{\mathrm{e}}^{\mathrm{h}} = [0^{o}\ 0^{o}]$ and the velocity to zero every 500 iterations. This ensures the eye orientation does not drift off so far that the eye never sees the hand. For each trajectory during testing, we initialize the eye velocity to zero and the eye position so that the gaze vector intersects the center of the palm.

We chose this method of random babbling and trajectory generation for its simplicity. There are a number of ways we can make the motion more biologically realistic, e.g., through the use of dynamic movement primitives (Schaal, 2006) for trajectory generation, or through the use of goal babbling (von Hofsten, 2004) to choose the via points. The use of dynamic movement primitives would alter the statistics of the image motion induced by the hand, which might change the smooth pursuit performance. The use of goal babbling might improve the speed of learning (Baranes and Oudeyer, 2013). These would be interesting extensions of the model to investigate. However, we do not expect their incorporation to change the main qualitative findings we report here.

For the sake of simplicity in our simulations, we use the iCub robot simulator to take into account the kinematics of the iCub robot as well as to model the geometry and appearance of the visual environment. We do not take into account the dynamics of the robot, nor do we incorporate a biologically realistic model for the eye movement dynamics. Rather, in each iteration, we move the robot arm to the configuration determined by the random babbling arm motion via position control. We assume that the eye velocity command is executed perfectly, by determining the location of the eye in the next iteration as the sum of the encoder measurement of the current position plus the velocity command in Equation (1), and move the eye there via position control. The images taken by the iCub in the new arm/eye positions determine the next visual input to the model. The proprioceptive input is determined from the motor encoders and their first and second differences in time. We believe that incorporating more realistic models of arm and eye dynamics are a natural next step. If these models are more biologically realistic, the model may give a better quantitative account of the performance of human subjects.

RESULTS

Learned Visual Representation

The basis vectors in the first stage GASSOM are analogous to the receptive fields of orientation-tuned simple cells in the human primary visual cortex (Chandrapala and Shi, 2015). As shown in **Figures 4A,B**, the basis vectors of the first GASSOM are tuned to specific orientations and spatial frequencies. The basis vectors corresponding to fine and coarse scales have similar

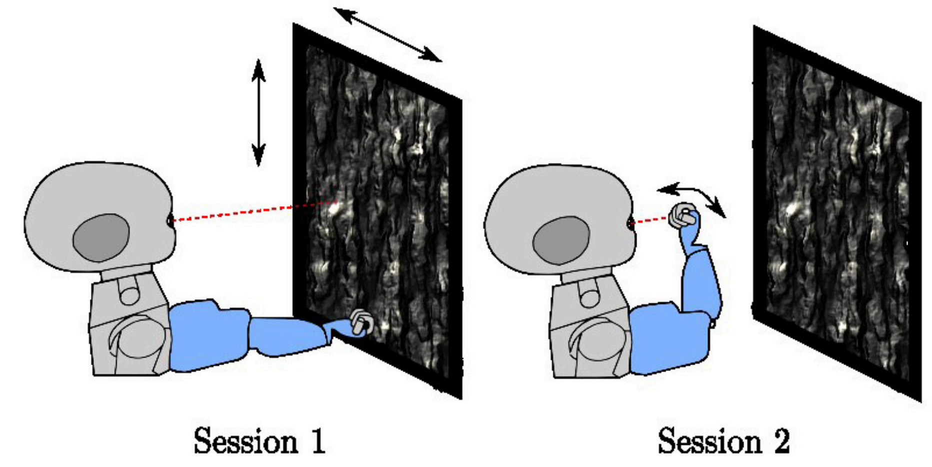


FIGURE 3 | The two-session types used in training, In session type 1, the arm rests in a fixed position outside the field of view of the robot while a textured plane moves in the background. In session type 2, the arm babbles in front of the robot with the textured plane remains stationary.

characteristics. The corresponding basis vectors associated with a node in the first GASSOM have a phase difference close to 90^o to represent the two orthogonal basis vectors. We initialize these basis vectors with the basis vectors learned on natural images. In addition, the parameters corresponding to the first GASSOM are fixed during the visual representation learning.

The prediction module predicts the projections at time t of the first stage GASSOM given the projections at time $t-1$ and the proprioceptive inputs. We evaluate the accuracy of the prediction module using the average cosine similarity between $\hat{\mathbf{p}}_{1,s,n,i}(t)$ and $\mathbf{p}_{1,s,n,i}(t)$. For this test, the eye and the arm are moved independently of each other. The eye velocities are sampled from Gaussian distributions fitted to the distribution of the eye velocity during training in both pan and tilt directions. The standard deviations of the fits are $\sigma_{\dot{\theta}_{\text{e,pan}}} = 0.7337$ deg/sample ($r^2 = 0.982$) and $\sigma_{\dot{\theta}_{\text{e,tilt}}} = 0.6387$ deg/sample ($r^2 = 0.989$). Arm trajectories are generated in a similar way to the training. In order to maintain the gaze on the robot hand, we execute a saccade to bring the eye gaze back to the center of the hand once the eye gaze drifts outside of the arm region. We evaluate the predictors over 10,000 iterations. The basis vectors from the subspaces with the highest and lowest average cosine similarity for two scales are outlined in green and blue in **Figures 4A,B**.

We fit a two-dimensional Gabor function to each basis vector to identify the factors influencing the prediction accuracy. **Figures 4C,D** show that the average cosine similarity of a predictor is related to the spatial frequency ($\frac{1}{\lambda}$; where λ is the spatial wavelength) of the basis vector. Higher spatial frequencies have a lower cosine similarity for predictors in both fine and coarse scales. Intuitively, basis vectors with higher spatial frequencies are more sensitive to retinal motion than those with low spatial frequencies. The transformations associated with higher spatial frequency basis vectors are more difficult to predict. The cluster of data points close to $\frac{1}{\lambda} = 0$ in **Figures 4C,D** are basis vectors whose fitted Gabor functions had very long spatial wavelengths. These typically corresponded to basis vectors with main support near the edges of the patch.

The second stage GASSOMs jointly encode the actual and predicted projections onto the subspaces of the first GASSOM. If the predictions from knowledge of the proprioception are accurate, there should be little difference between the actual and predicted projections. Differences between the two arise due to exteroceptive motion, which cannot be predicted from proprioception, as well as inaccuracy in the predictor. The basis vectors in the second GASSOM encode these differences and inherit orientation and spatial frequency tuning from the first stage. Encoding only the residual motion after prediction helps to reduce correlation between the visual and proprioceptive cues.

We examine the tuning of the basis vectors in the second GASSOM using drifting two-dimensional cosine gratings in 10×10 pixel image patches. For this test, we fix the proprioceptive input to zero self-motion. We record the responses from the subspaces of the second GASSOM to all combinations of motion, spatial frequency and orientation, where spatial wavelengths varied from 3 to 20 pixels, motion from -2 to 2 deg/sample, orientation from 0° to 180°. For each subspace, we determine the preferred tuning from the combination that resulted in the maximum magnitude response.

The tuning characteristics for the fine scale basis vectors are provided in **Figures 5A,B**. Here, we also present the tuning statistics corresponding to a model without the prediction module shown in **Figures 5C,D**. The majority of the basis vectors are tuned to zero velocity for both architectures as shown in **Figures 5A,C**. The tuning velocities of the architecture with, without prediction have a variance of 0.3660, 0.4286 $(deg/sample)^2$. The tuning orientations are distributed close to a uniform distribution as shown in **Figures 5B,D**. The KL divergence with the uniform distribution of orientations for the architecture with, without prediction is 0.0113, 0.0397. Hence, the two architectures prefer zero retinal slip for all the tuning

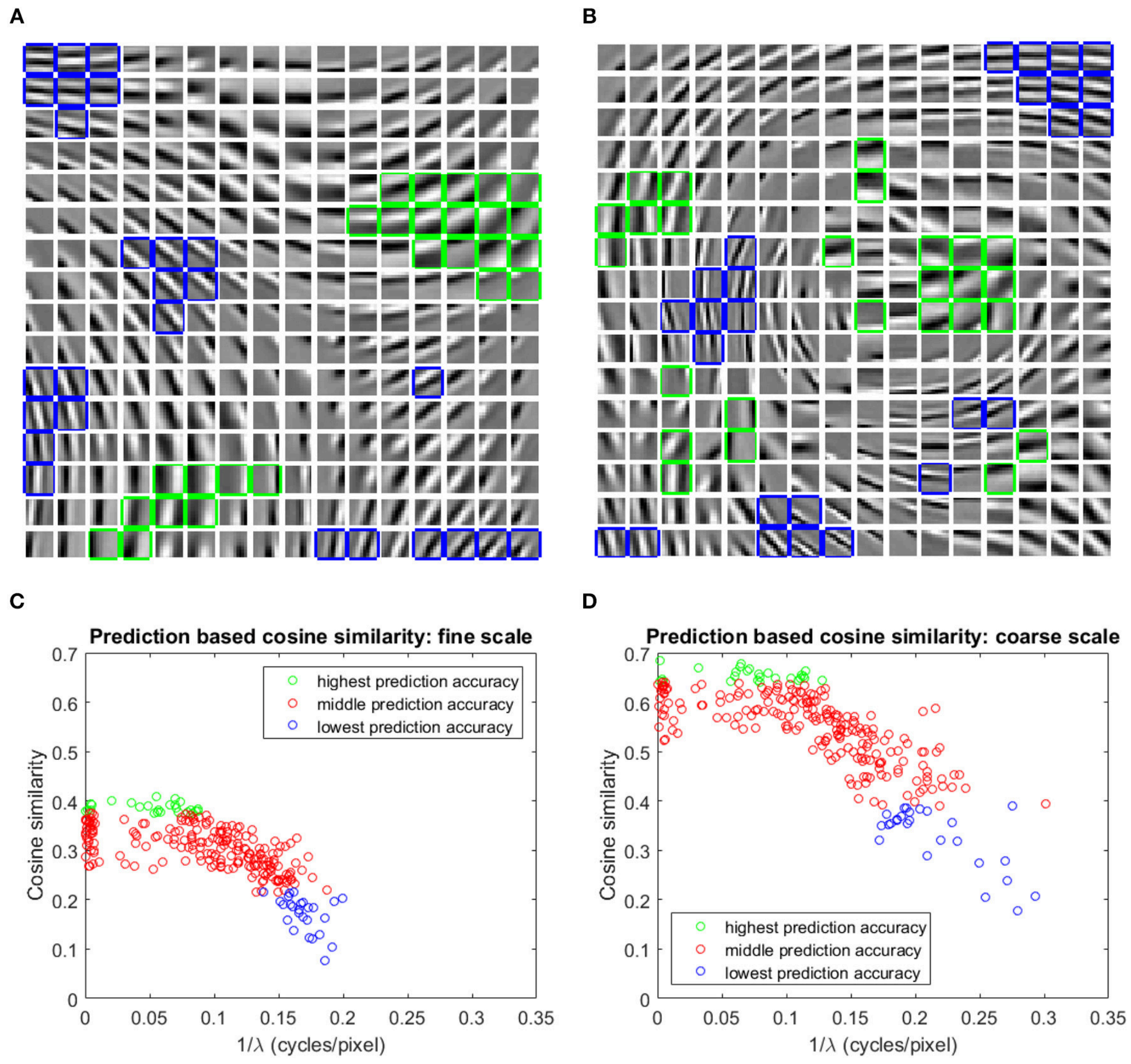


FIGURE 4 | (A,B) One out of each pair of basis vectors spanning the 256 subspaces in the fine **(A)**, and coarse **(B)** scale GASSOMs. The basis vectors are shown as a **16** × **16** array of **10** × **10** images. In each subplot, the twenty six basis vectors of the subspaces with the highest (lowest) average cosine similarity between the predicted and actual outputs are outlined in green (blue). **(C,D)** Scatter plots of the average cosine similarity of the prediction vs. best-fit spatial frequency for the fine **(C)**, and coarse **(D)** scale basis vectors.

orientations. For the architecture with prediction, the variance of the tuning velocities is lower compared to the other architecture.

Learning the Eye Motor Controller

The eye controller is tested by evaluating the tracking performance for a set of 10 different arm trajectories, each lasting 1,000 iterations. The trajectories are generated in the same way as in training. The performance is measured by computing the root mean squared error (RMSE) between the target and actual eye velocities.

We compare the performance of the system in three different scenarios. In all cases, the robot attempts to track the end effector of the robot. In the first case, the model is driven by both visual and proprioceptive stimuli. In the second case, the model is purely driven by vision, with illusory proprioceptive input being provided that suggests that the arm is fixed at the resting position outside the field of view used in training session type 1. In the third case, the model is purely driven by proprioception, with the visual feature $\boldsymbol{\phi}_v(t)$ replaced by the expected value computed over time $E[\boldsymbol{\phi}_v(t)\ ]$. This approach allows us to compare the three cases by providing the same visual stimuli (the robot hand) in distinct scenarios. **Figure 6** depicts the learning progress recorded at 6 checkpoints occurring every 80,000 iterations. The RMSE for both pan and tilt angular velocities are averaged over 30 different trials comprising 10 trajectories and 3 different training trials. The learning curves in **Figure 6** illustrate that in steady state, using both visual and proprioceptive stimuli is much more accurate than using either stimuli alone. This is typical of multimodal integration.

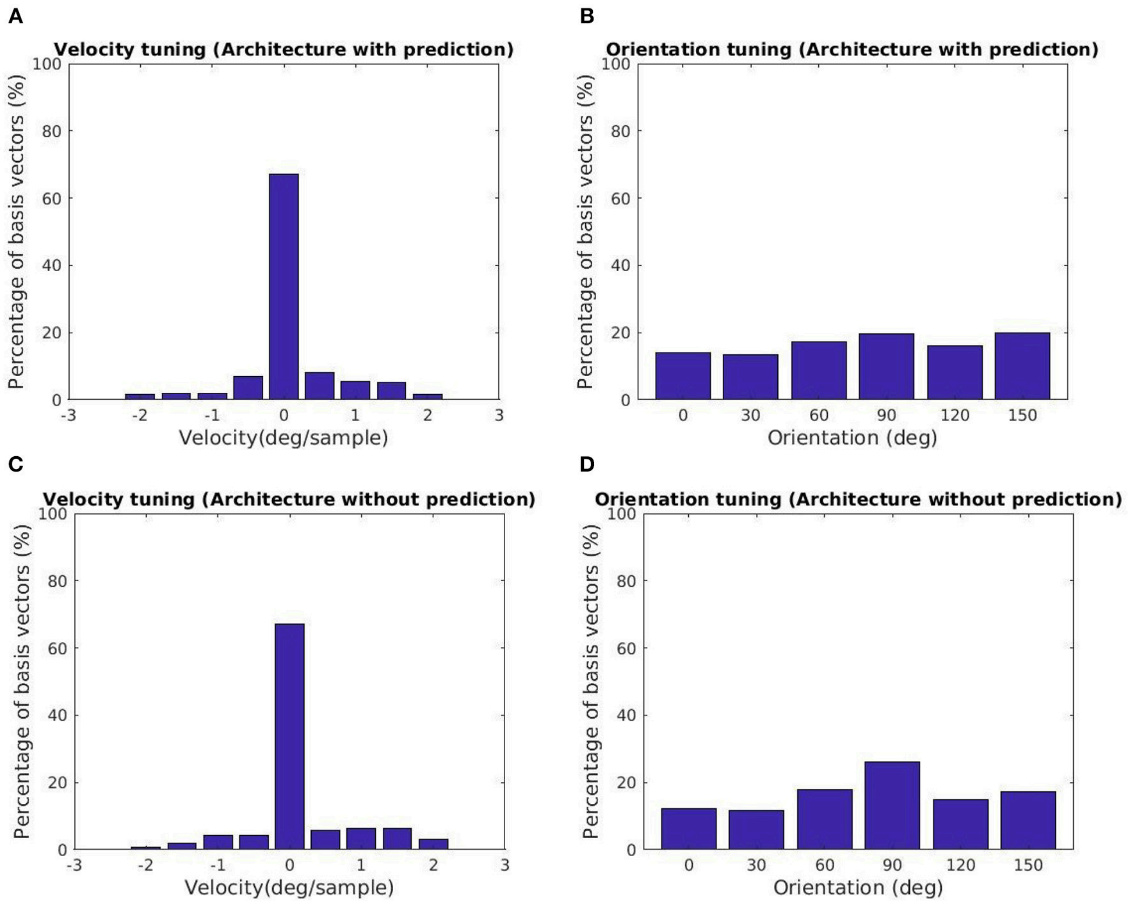


FIGURE 5 | Tuning statistics for the second stage GASSOM. **(A,B)** (Velocity, Orientation) statistics for the architecture with the prediction module. **(C,D)** (Velocity, Orientation) statistics for the architecture without prediction. The majority of the basis vectors in the second GASSOM are tuned in to zero retinal slip in both cases.

Comparison With Psychophysical Experiments

In this section, we compare the eye movements generated by the model to that of the human oculomotor system using an experimental protocol similar to that described in Vercher et al. (1993), which is illustrated in **Figure 7**. Vercher et al. measured the frequency responses of the human oculomotor plant during visual tracking for five subjects (4 males and 1 female) in two different cases. In the first case, the subject was tracking a target moving in a sinusoidal trajectory as shown in **Figure 7A**. In the second case, the target trajectory was controlled by the subject using his/her arm while tracking the corresponding target with the eyes as shown in **Figure 7B**. The eye movements in these two cases were compared to understand the role of proprioception in oculomotor control.

In our experiment, with the model to generate comparable data, we use the end effector of the robot arm as the target. The end effector is moved in a sinusoidal trajectory between -6^o and $+6^o$ in the pan direction with respect to the eye. The frequency of the motion varies from 0.5 to 2 Hz in 0.5 Hz steps. Sample trajectories generated by the model at 1 Hz frequency are shown in **Figure 8**. We fit a sine function to the eye velocity to compute the velocity gain and phase difference with reference to the target trajectory. The eye trajectory in **Figure 8A** shows a higher gain and a lower phase delay compared to the trajectory in **Figure 8B**. Hence, the addition of proprioceptive information improves the velocity gain in comparison to vision alone. The addition of proprioception also reduces the phase delay. Our system is also able to move the eye solely with the proprioceptive input as illustrated in **Figure 8C**.

We compare the results of our model with human performance, by extracting the frequency response data from Figures 2, 3 in Vercher et al. (1993), which show the eye velocity gains and phase delays averaged across the five subjects. **Figure 9** compares the frequency responses. The model and human data have qualitative similarities. According to the gain plots in **Figures 9A,C**, both responses have a higher gain in the presence of both vision and proprioception. In addition, proprioception combined with vision has a lower phase delay as shown in **Figures 9B,D**. For the subjects in this study, both efferent and afferent information is available during tracking of the self-moved target, whereas our model only includes afferent information from proprioception. Vercher et al. (1996) studied the role of proprioception in eye-hand coordination

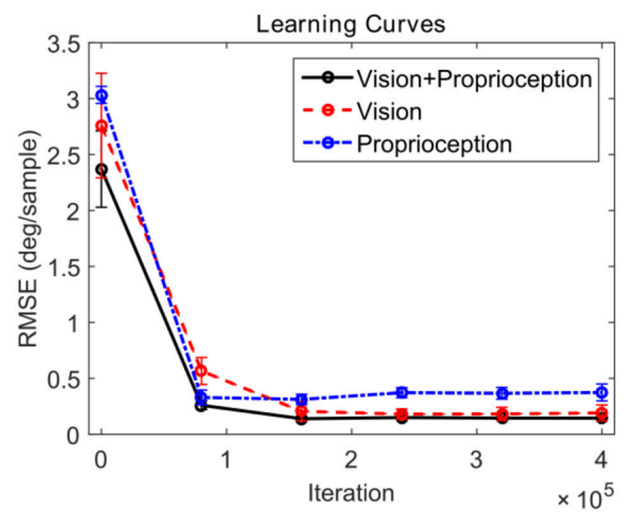


FIGURE 6 | Learning curves corresponding to three testing scenarios. The performance for pan and tilt angular velocities are averaged. The black solid line shows the learning curve corresponding to the system using both visual and proprioceptive inputs to track the robot hand. The red dashed line shows the learning curve corresponding to the system using the pure visual input to track the robot hand. The blue dash-dot line shows the learning curve corresponding to the system using pure proprioceptive input.

by comparing the behavior between deafferented and control subjects. The deafferented subjects showed little difference in performance between tracking a self-moved target and an external target. This highlights the importance of proprioception as a non-visual signal for smooth pursuit control.

Qualitative Evaluation of Performance

To observe the tracking performance qualitatively, we generate a video comparing the eye trajectories as the arm moves randomly for the three different combinations of cues[1] This video demonstrates the performance by projecting the gaze position to an image frame obtained from a fixed camera for three different testing cases. The right eye camera is moving differently in all three testing cases. Relating the gaze to a reference camera frame makes it easier to compare the different cases. From the video, the pure proprioception based tracking underperforms in comparison to the other two cases. This is consistent with the velocity gain and phase delay responses in **Figure 9**.

The performance in the video can be summarized by projecting the gaze vectors to the end effector coordinate system. The origin of this coordinate system is located on the palm of the robot end effector. **Figure 10** shows distributions of the intersections between the eye gaze direction and the plane passing through the origin parallel to the palm surface over the entire tracking trajectory as two-dimensional heat maps, one for each of the testing cases. In comparison to **Figures 10A,B, 10C** shows larger variability in the gaze position on the end effector. This is consistent with the poorer tracking performance for proprioception alone than in the other two cases. The model does not explicitly define a precise end effector location to be tracked. Since only acceleration commands are generated, the gaze position can drift. Thus, the mean gaze position projected to the palm varies across different training trials for the three different testing cases.

We also illustrate the system's robustness to changes in the visual appearance of the robot end effector qualitatively through an accompanying video[2] We change the appearance by moving the wrist and fingers of the iCub with sinusoidal joint trajectories. No information about finger and wrist motion is provided to the system. When eye acceleration is driven by vision alone, the eye drifts away faster from the end effector. The changes in the appearance of the end effector introduce visual perturbations, which are challenging to follow in comparison to changes due to translation of the end effector. For the other two cases, the proprioceptive inputs, which are not altered due to the changes in the visual appearance of the robot end effector, enable the robot to maintain the gaze on the palm for a longer time.

Importance of the Prediction Module

To identify the role of the prediction module in the proposed model, we compare the performance of three different models. **Figure 11A** shows a general architecture that covers the three models compared in this study. The three models differ only in their visual pathways. The proprioceptive features and the eye controller neural network are identical. **Figure 11B** illustrates the prediction based visual pathway proposed in this paper. The pathway in **Figure 11C** has a very similar structure, except that the prediction module is removed. Comparing these two models enables us to identify the benefit of the prediction module. The pathway shown in **Figure 11D** is the sparse coding based visual representation used in our earlier work (Wijesinghe et al., 2017).

To compare the performance, each model is tested for 20 different testing trials (10 different trajectories for each of the two training scenario types). **Figure 12** shows the RMSEs between the target and actual eye velocities. We perform paired sample t-tests to compare the RMSE of the model with prediction with the RMSEs of the other two. The effect size is also computed according to the Cohen's d formula. The Differences between the performance of the models in **Figures 11B,C** are not statistically significant for vision and proprioception ($p = 6.27 \times 10^{-1}$, effect size $= -0.0876$) nor vision alone ($p = 4.40 \times 10^{-1}$, effect size$= 0.1619$). The model with prediction performs significantly better for proprioception alone ($p = 6.16 \times 10^{-10}$, effect size $= 4.2334$). We have similar findings for the Sparse Coding based model in **Figure 11D**. Differences in performance are not significant for vision and proprioception ($p = 7.29 \times 10^{-2}$, effect size $= 0.6531$) nor vision alone ($p = 1.49 \times 10^{-1}$, effect size $= -0.4611$). The model with prediction performs significantly better with proprioception alone ($p = 5.94 \times 10^{-15}$, effect size $= 4.7588$).

[1]The video is available online at https://youtu.be/PPb7KwWefBI

[2]The video is available online at https://youtu.be/RsGTmbb0cf0

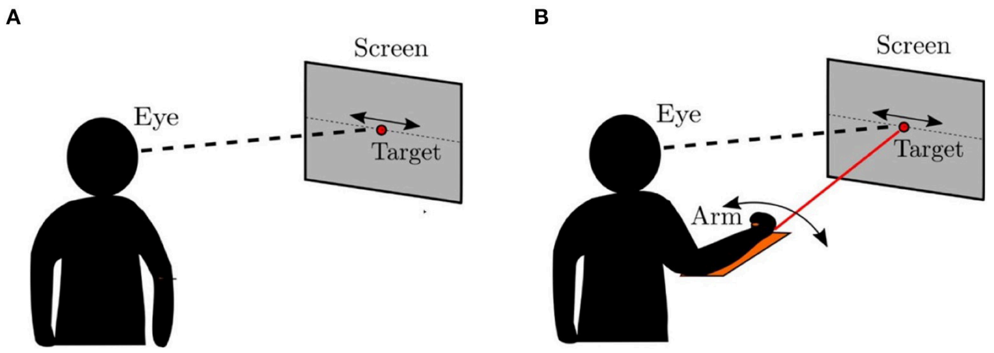


FIGURE 7 | Psychophysical experiment (Vercher et al., 1993). **(A)** The target was controlled externally. The subject was asked to track the target using direct visual input. **(B)** The target was controlled by the subject using a lever. The subjects simultaneously tracked the target ulitlizing both visual and proprioceptive inputs.

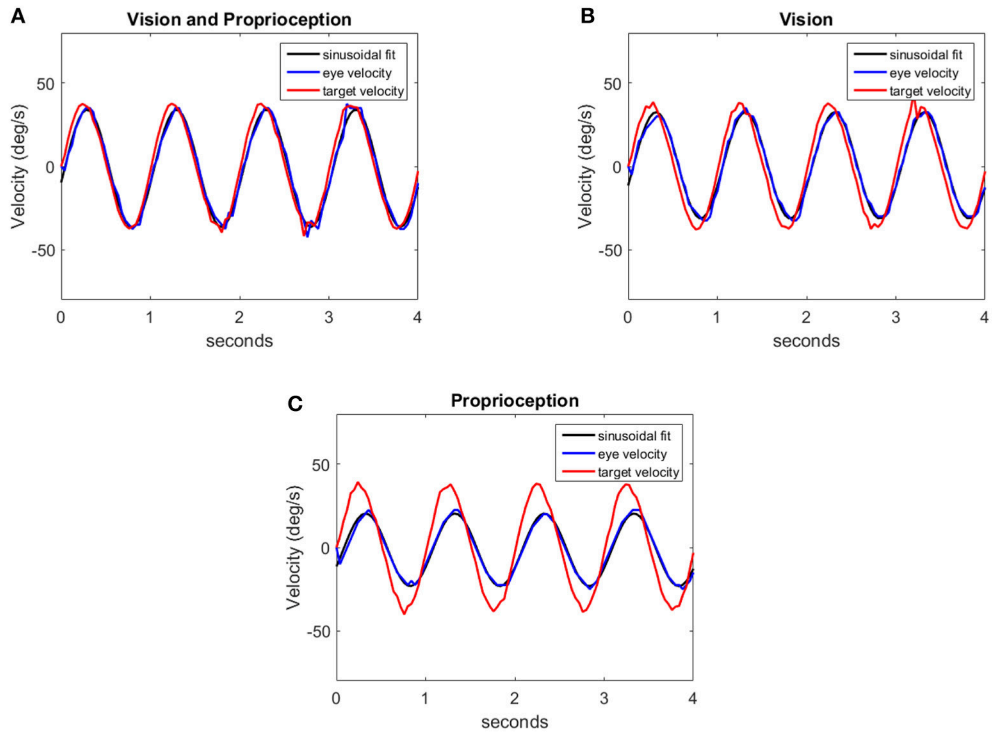


FIGURE 8 | Eye velocity trajectories corresponding to the eye responses for different sensory modalities: **(A)** Vision and Proprioception, **(B)** Vision, **(C)** Proprioception. Each figure contains a trajectory (red) indicating the target velocity, the respective eye velocity (blue) and the sinusoidal fit (black) for the eye velocity.

These results demonstrate that the proposed prediction based model exhibits superior performance compared to models without prediction. We attribute this to reduced correlation between the visual and proprioceptive features in the prediction-based model.

DISCUSSION

In this article, we propose a model based on the Active Efficient Coding (AEC) framework that enables a robot to learn to track its end effector using a combination of visual and proprioceptive cues. Rather than simply concatenating the two sets of features, the proposed model predicts visual consequences of actions, which removes information correlated with proprioception from vision. The model enables a robot to learn to track an object for three cases: using both visual and proprioceptive cues corresponding to the typical case of end effector tracking, using only visual cues corresponding to tracking of an external independently moving object, and using pure proprioception corresponding to tracking of the end effector in darkness.

The incorporation of prediction is motivated by recent studies on neural responses in V1 of mice during locomotion (Niell and Stryker, 2011; Keller et al., 2012; Attinger et al., 2017). These studies suggest that the responses of cells in the V1

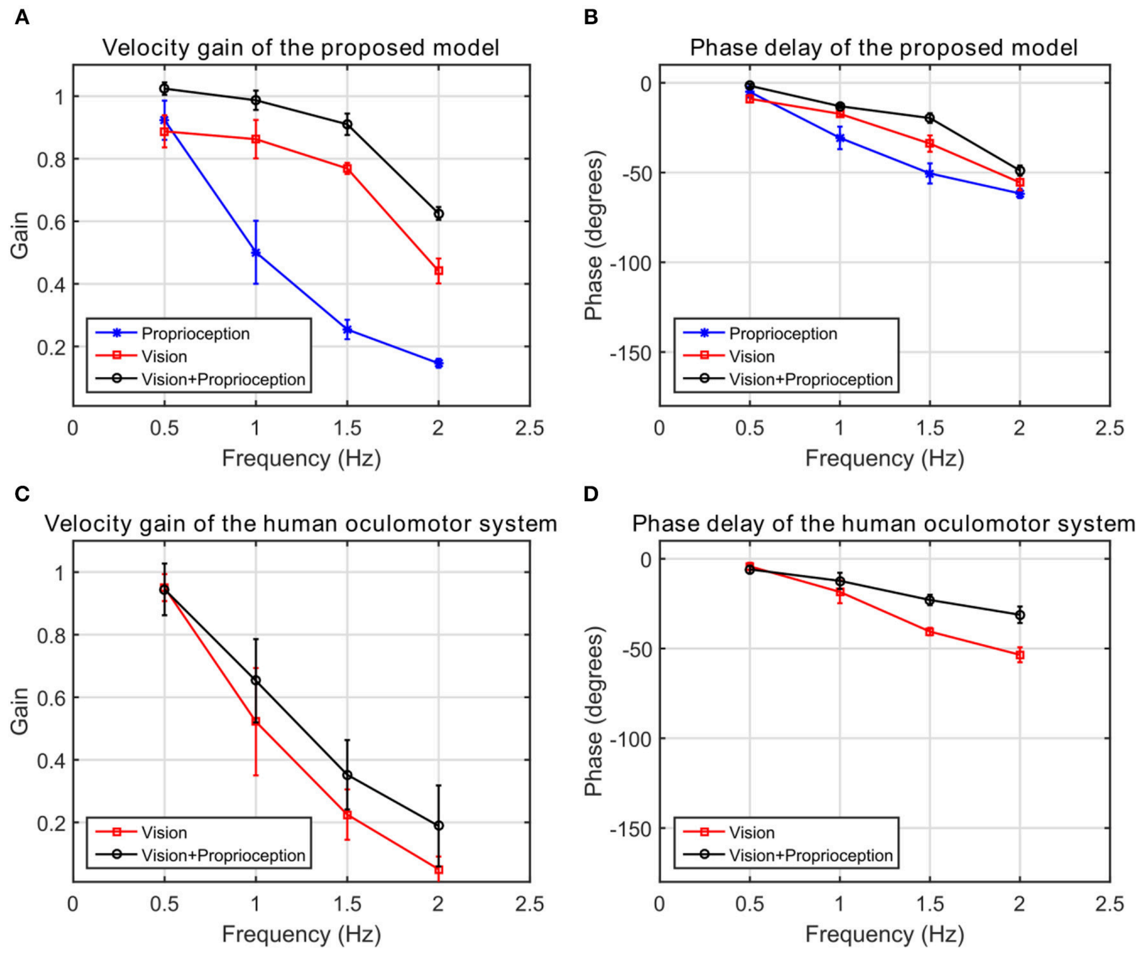


FIGURE 9 | The frequency response of the human oculomotor system compared with the frequency response of the proposed model. **(A)** The velocity gain of the proposed system. **(B)** The phase delay of the proposed system. **(C)** The velocity gain of the human oculomotor system. **(D)** The phase delay of the human oculomotor system.

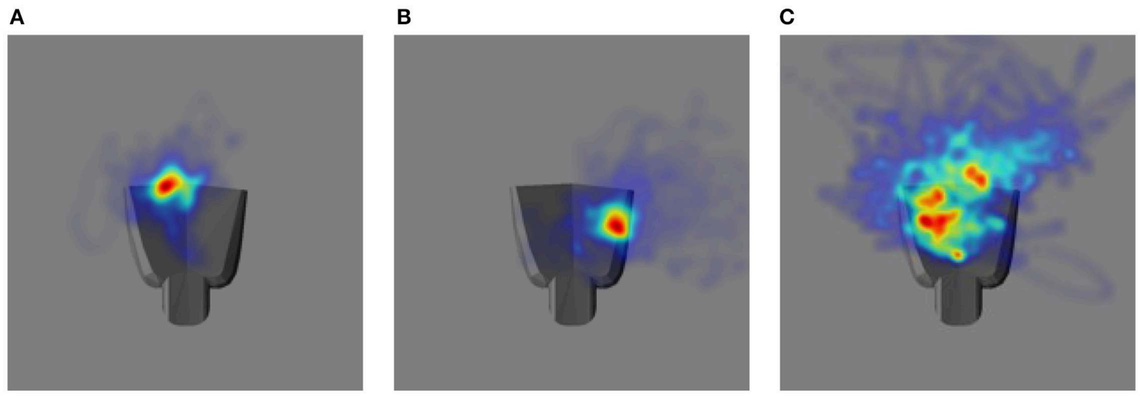


FIGURE 10 | The heat map illustrating the distribution of eye gaze intersection with a plane parallel to the palm of the robot hand. The robot hand is superimposed to each image to have a qualitative comparison: **(A)** Vision and Proprioception **(B)** Vision **(C)** Proprioception.

depend upon predictions of the sensory consequences of motor actions. In fact, locomotion improves the encoding of visual stimuli (Dadarlat and Stryker, 2017). In our model, the visual representation encodes the residual motion after removing the predicted effects of self-motion from the observed visual flow. We show that the proposed prediction module has the ability

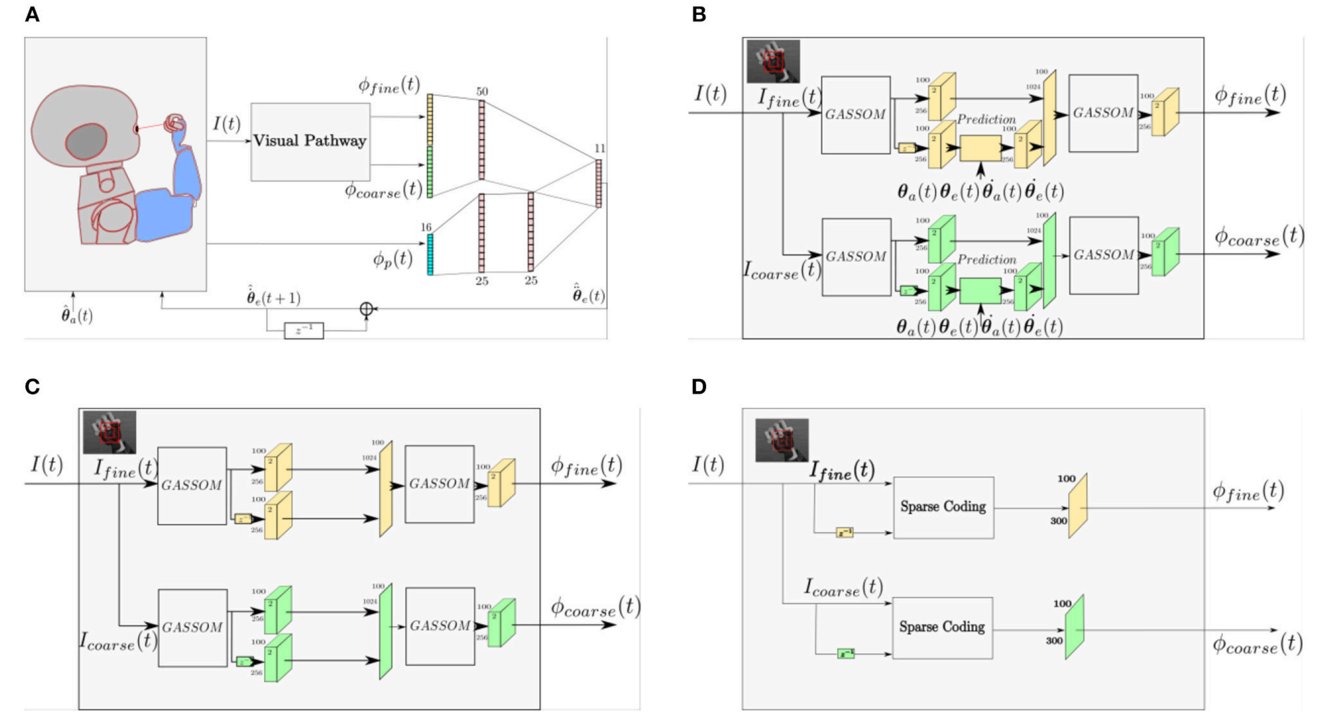


FIGURE 11 | (A) A general architecture representing all of the architectures compared in this study. **(B)** The prediction based visual pathway proposed in this paper. **(C)** The visual pathway without the prediction module. **(D)** A Sparse Coding based visual pathway.

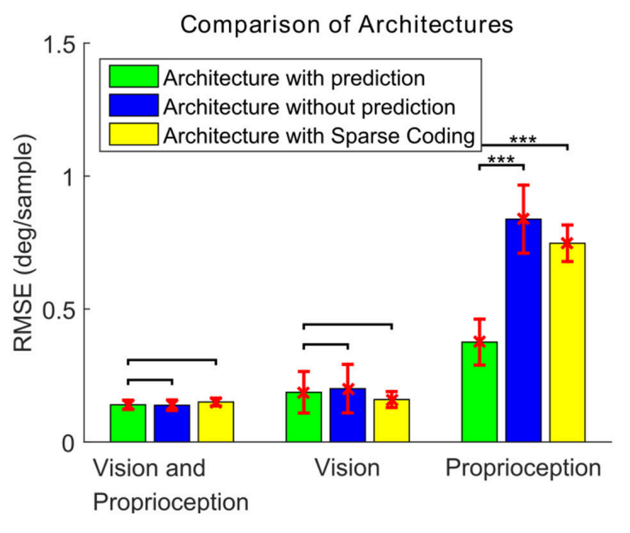


FIGURE 12 | The RMSE (mean ± standard deviation) of the eye velocity compared to a target velocity for three different architectures and three different combinations of sensory modalities. For the paired *t*-test comparisons, no mark indicates differences are not statistically significant ($p >$ 0.05), ***indicates statistical significance ($p < 0.001$).

to predict the visual sensory consequences of proprioceptive inputs (**Figure 4**). Specifically, the visual sensory consequences for subspaces with low spatial frequency basis vectors in the first stage GASSOM are easier to predict compared to higher spatial frequencies.

The incorporation of learning into both the perception and action components of the perception-action loop in this model allows the sensory representation and the action generation network to co-adapt as the agent behaves in the environment. We characterized the performance of the eye controller both quantitatively and qualitatively. Using both visual and proprioceptive sensory stimuli to drive eye motion results in more accurate tracking of the end effector compared to using either sensory stimulus alone. Moreover, the inclusion of proprioception also makes the model more robust to changes in the appearance of the end effector.

We compare predictions of our model with findings from human psychophysical experiments studying the contribution of proprioception to human oculomotor control. Our results in **Figure 9B** suggest that incorporating proprioception reduces phase delay. This characteristic of the human oculomotor system has been found repeatedly. First, early work analyzing self-moved targets showed that the information about arm motion plays an important role in self-motion tracking (Steinbach and Held, 1968). The eye motion lagged behind target motion less for active arm motions than passive arm motions. Second, active and passive hand motion tracking along with tracking of external visual targets were qualitatively compared in Mather and Lackner (1980). External target tracking used a larger number of saccades per cycle and had larger latency compared to the oculomotor tracking of the hand. The external target tracking was more challenging since the target motion was unpredictable. Third, experiments conducted by Gauthier et al. (1988) quantitatively compared eye tracking of an external object and of the subject's hand. The average latency when the eye tracked an external object (150 ± 30 ms) was much longer than the average latency when the eye tracked the hand (30 ± 10 ms). This delay was also prominent in the onset of smooth pursuit eye motions (Domann et al., 1989). Finally, Chen et al. (2016) showed that eye precedes a target controlled by the finger in congruent pursuit in comparison to opposite movements. However, our model does not exhibit this property.

Our results qualitatively agree with multiple psychophysical studies showing that proprioception can also change the velocity gain of the human oculomotor plant. During eye tracking of an external object, the eye velocity saturated at very low velocities around 40 deg/s compared to 100 deg/s during eye tracking of the hand (Domann et al., 1989). As shown in **Figure 9A**, velocity gains are larger at higher frequencies when the robot tracks its hand compared to an external object. Our model demonstrates that with the assistance of proprioception, the model is capable of maintaining a higher tracking gain for high frequency stimuli. This property is qualitatively consistent with the human oculomotor system (Vercher et al., 1993).

Human eye motion is mainly driven by visual stimuli. However, non-visual signals can also drive eye motion in certain tasks (e.g., in the darkness). **Figure 8C** illustrates that for our model proprioception alone has the capability to elicit eye movements. Several articles have studied the contribution of non-visual signals to oculomotor control. First, Steinbach (1969) showed that proprioceptive inputs alone were sufficient to generate smooth pursuit eye movements. Second, smooth pursuit movements generated by non-visual stimuli were studied in Berryhill et al. (2006). Tracking a pendulum in the darkness, proprioceptive stimuli had a velocity gain close to 0.3. On the other hand, with direct visibility of the pendulum, the subject had the ability to track the pendulum more accurately with a velocity gain of 0.7. Our model exhibits similar qualitative changes in visual vs. proprioceptive gain.

Although our model exhibits similar qualitative characteristics as human oculomotor tracking, it does not match quantitatively. In particular, the velocity gain of the model is much higher than that of the human oculomotor system (**Figure 9**). This mismatch may arise because we assume in our simulations that the eye velocity command generated by the model is executed perfectly by the eye. A more realistic model would include processing and propagation delays, a more realistic model of the neural control of eye velocity, as well as a dynamical model of the physical oculomotor plant. For example, the cerebellum is a part of the neural circuit for eye-hand coordination for oculomotor control (Miall et al., 2001), but this is not reflected in the proposed model. We anticipate that the incorporation of these elements into future extensions of the model would degrade the velocity gain observed during tracking, bringing the model simulations into closer quantitative agreement with human performance.

This work can be extended in several directions. First, the model could be extended to include saccadic eye movements. Tracking targets with the eyes typically consists of a combination of pursuit and saccadic eye movements. For example, the majority of the eye movements when humans tracking the arm in darkness are saccades (Dieter et al., 2014).

Second, new sensory modalities might be added to the proposed model. The proposed model is only a first step toward integrating cross-sensory prediction. While a straightforward extension might be to add additional inputs to the predictor network, consideration of the problem of adding new sensory modalities raises a number of intriguing questions. For example, which sensory inputs should be predicted from which others? Here we have considered prediction in only one direction. Another question is how to deal with the different possible combinations of sensory cues that might be available.

Third, the arm trajectory generation could be made more biologically realistic, e.g., through the use of dynamical movement primitives for trajectory generation (Schaal, 2006), or through the use of goal babbling to choose the via points (von Hofsten, 2004). In this work, we used random babbling and trajectory generation for its simplicity. The use of dynamic movement primitives would alter the statistics of the image motion induced by the hand, which might change the smooth pursuit performance, e.g., the final steady state error in **Figure 6** or the shape of the frequency response curves in **Figure 9**. The use of goal babbling might improve the speed of learning (Baranes and Oudeyer, 2013). These would be interesting extensions of the model to investigate. However, we do not expect that their incorporation to change the main qualitative findings we report here, e.g., the ordering of the degradation in performance and proprioceptive or visual cues are removed.

Finally, the model might be integrated into a more comprehensive framework for hand-eye coordination that would include other tasks, such as reaching. In a sense, reaching is the inverse of the problem studied here. Our model generates commands to change eye gaze based on visual information and arm motion. Visually guided reaching involves the generation of an arm motion command to move the end effector to a visual target. The required mappings, e.g., between gaze direction and/or visual target location and end effector motion are often learned using a motor babbling process similar to that used here, where the end effector is tracked as the arm moves (Burger et al., 2018). In other cases, gaze is controlled to bring the target object or end effector to the image center (Huelse et al., 2010; Jamone et al., 2012; Savastano and Nolfi, 2013). In these works, the problem of tracking the end effector/target was simplified by attaching a marker to the end effector/target. Since our model does not require explicit markers, it might be used to relax some of the assumptions made by prior work. Because the location of the eye gaze drifts on the palm area (**Figure 10**), gaze direction does not directly correspond to a specific position on the end effector in trajectories generated by the model. However, the model could be used to generate data to learn an approximate mapping between gaze direction and end effector position.

AUTHOR CONTRIBUTIONS

All authors participated in designing experiments and writing the paper. LW conducted the experiments.

FUNDING

This work was supported in part by the General Research Fund of the Hong Kong Research Grants Council, under grant 16209014 (LW, BS), Johanna Quandt Foundation (JT), and the European Union's Horizon 2020 research and innovation program under grant 713010 (JT).

REFERENCES

Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., et al. (2009). Cognitive developmental robotics: a survey. *IEEE Trans. Auton. Ment. Dev.* 1, 12–34. doi: 10.1109/TAMD.2009.2021702

Attinger, A., Wang, B., and Keller, G. B. (2017). Visuomotor coupling shapes the functional development of mouse visual cortex. *Cell* 169, 1291–1302. doi: 10.1016/j.cell.2017.05.023

Baranes, A., and Oudeyer, P. Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Rob. Auton. Syst.* 61, 49–73. doi: 10.1016/j.robot.2012.05.008

Bennett, D. J., Geiger, D., and Hollerbach, J. M. (1991). Autonomous robot calibration for hand-eye coordination. *Int. J. Rob. Res.* 10, 550–559. doi: 10.1177/027836499101000510

Berryhill, M. E., Chiu, T., and Hughes, H. C. (2006). Smooth pursuit of nonvisual motion. *J. Neurophysiol.* 96, 461–465. doi: 10.1152/jn.00152.2006

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. (2009). Natural actor–critic algorithms. *Automatica* 45, 2471–2482. doi: 10.1016/j.automatica.2009.07.008

Broun, A., Beck, C., Pipe, T., Mirmehdi, M., and Melhuish, C. (2014). Bootstrapping a robot's kinematic model. *Rob. Auton. Syst.* 62, 330–339. doi: 10.1016/j.robot.2013.09.011

Burger, W., Wieser, E., Dean-Leon, E., and Cheng, G. (2018). "A scalable method for multi-stage developmental learning for reaching," in *IEEE International Conference on Development and Learning and on Epigenetic Robotics* (Lisbon), 60–65. doi: 10.1109/DEVLRN.2017.8329788

Chandrapala, T. N., and Shi, B. E. (2015). Learning slowness in a sparse model of invariant feature detection. *Neural Comput.* 27, 1496–1529. doi: 10.1162/NECO_a_00743

Chen, J., Valsecchi, M., and Gegenfurtner, K. R. (2016). Role of motor execution in the ocular tracking of self-generated movements. *J. Neurophysiol.* 116, 2586–2593. doi: 10.1152/jn.00574.2016

Corke, P. (2017). *Robotics, Vision and Control*. Cham: Springer. doi: 10.1007/978-3-319-54413-7

Dadarlat, M. C., and Stryker, M. P. (2017). Locomotion enhances neural encoding of visual stimuli in mouse V1. *J. Neurosci.* 37, 3764–3775. doi: 10.1523/JNEUROSCI.2728-16.2017

Dalal, N., and Triggs, B. (2005). "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition* (San Diego, CA), 886–893,doi: 10.1109/CVPR.2005.177

Dieter, K. C., Hu, B., Knill, D. C., Blake, R., and Tadin, D. (2014). Kinesthesis can make an invisible hand visible. *Psychol. Sci.* 25, 66–75. doi: 10.1177/0956797613497968

Domann, R., Bock, O., and Eckmiller, R. (1989). Interaction of visual and non-visual signals in the initiation of smooth pursuit eye movements in primates. *Behav. Brain Res.* 32, 95–99. doi: 10.1016/S0166-4328(89)80077-4

Fantacci, C., Pattacini, U., Tikhanoff, V., and Natale, L. (2017). "Visual end-effector tracking using a 3D model-aided particle filter for humanoid robot platforms," in *IEEE International Conference on Intelligent Robots and Systems* (Vancouver, BC), 1411–1418. doi: 10.1109/IROS.2017.8205942

Gatla, C. S., Lumia, R., Wood, J., and Starr, G. (2007). An automated method to calibrate industrial robots using a virtual closed kinematic chain. *IEEE Trans. Robot.* 23, 1105–1116. doi: 10.1109/TRO.2007.909765

Gauthier, G. M., and Mussa Ivaldi, F. (1988). Oculo-manual tracking of visual targets in monkey: role of the arm afferent information in the control of arm and eye movements. *Exp. Brain Res.* 73, 138–154. doi: 10.1007/BF002 79668

Gauthier, G. M., Vercher, J. L., Mussa Ivaldi, F., and Marchetti, E. (1988). Oculo-manual tracking of visual targets: control learning, coordination control and coordination model. *Exp. Brain Res.* 73, 127–137. doi: 10.1007/BF00279667

Hersch, M., Sauser, E., and Billard, A. (2008). Online learning of the body schema. *Int. J. Humanoid Robot.* 5, 161–181. doi: 10.1142/S0219843608001376

Hoffmann, M., Marques, H., Arieta, A., Sumioka, H., Lungarella, M., and Pfeifer, R. (2010). Body schema in robotics: a review. *IEEE Trans. Auton. Ment. Dev.* 2, 304–324. doi: 10.1109/TAMD.2010.2086454

Hollerbach, J. M., and Wampler, C. W. (1996). The calibration index and taxonomy for robot kinematic calibration methods. *Int. J. Rob. Res.* 15, 573–591. doi: 10.1177/027836499601500604

Huelse, M., McBride, S., Law, J., and Lee, M. (2010). Integration of active vision and reaching from a developmental robotics perspective. *IEEE Trans. Auton. Ment. Dev.* 2, 355–367. doi: 10.1109/TAMD.2010.2081667

Jamone, L., Natale, L., Nori, F., Metta, G., and Sandini, G. (2012). Autonomous online learning of reaching behavior in a humanoid robot. *Int. J. Humanoid Robot.* 09:1250017. doi: 10.1142/S021984361250017X

Keller, G. B., Bonhoeffer, T., and Hübener, M. (2012). Sensorimotor mismatch signals in primary visual cortex of the behaving mouse. *Neuron* 74, 809–815. doi: 10.1016/j.neuron.2012.03.040

Kohonen, T. (1996). Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map. *Biol. Cybern.* 75, 281–291. doi: 10.1007/s004220050295

Lisberger, S. G. (2010). Visual guidance of smooth-pursuit eye movements: sensation, action, and what happens in between. *Neuron* 66, 477–491. doi: 10.1016/j.neuron.2010.03.027

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* 60, 91–110. doi: 10.1023/B:VISI.0000029664.99615.94

Mather, J. A, and Lackner, J. R. (1980). Visual tracking of active and passive movements of the hand. *Q. J. Exp. Psychol.* 32, 307–315. doi: 10.1080/14640748008401166

Miall, R. C., Reckess, G. Z., and Imamizu, H. (2001). The cerebellum coordinates eye and hand tracking movements. *Nat. Neurosci.* 4, 638–644. doi: 10.1038/88465

Niell, C. M., and Stryker, M. P. (2011). Modulation of visual responses by behavioral state in mouse visual cortex. *Neuron* 65, 472–479. doi: 10.1016/j.neuron.2010.01.033.Modulation

Olmos, A., and Kingdom, F. A. (2004). A biologically inspired algorithm for the recovery of shading and reflectance images. *Perception* 33, 1463–1473. doi: 10.1068/p5321

Rao, R. P., and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.* 2, 79–87. doi: 10.1038/4580

Rochat, P. (1998). Self-perception and action in infancy. *Exp. Brain Res.* 123, 102–109. doi: 10.1007/s002210050550

Saleem, A. B., Ayaz, A., Jeffery, K. J., Harris, K. D., and Carandini, M. (2013). Integration of visual motion and locomotion in mouse visual cortex. *Nat. Neurosci.* 16, 1864–1869. doi: 10.1038/nn.3567

Savastano, P., and Nolfi, S. (2013). A robotic model of reaching and grasping development. *IEEE Trans. Auton. Ment. Dev.* 5, 326–336. doi: 10.1109/TAMD.2013.2264321

Schaal, S. (2006). "Dynamic movement primitives -a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines,* eds H. Kimura, K. Tsuchiya, A. Ishiguro, and H. Witt (Tokyo: Springer-Verlag), 261–280. doi: 10.1007/4-431-31381-8_23

Schillaci, G., Hafner, V. V., and Lara, B. (2014). "Online learning of visuo-motor coordination in a humanoid robot. A biologically inspired model" in

IEEE International Conference on Development and Learning and on Epigenetic Robotics (Genoa), 130–136. doi: 10.1109/DEVLRN.2014.6982967

Srinivasan, M. V., Laughlin, S. B., and Dubs, A. (1982). Predictive coding: a fresh view of inhibition in the retina. *Proc. R. Soc. B Biol. Sci.* 216, 427–459. doi: 10.1098/rspb.1982.0085

Steinbach, M. J. (1969). Eye tracking of self-moved targets: the role of efference. *J. Exp. Psychol.* 82, 366–376. doi: 10.1037/h0028115

Steinbach, M. J., and Held, R. (1968). Eye tracking of observer-generated target movements. *Science* 161, 187–188. doi: 10.1126/science.161.3837.187

Sturm, J., Plagemann, C., and Burgard, W. (2009). Body schema learning for robotic manipulators from visual self-perception. *J. Physiol. Paris* 103, 220–231. doi: 10.1016/j.jphysparis.2009.08.005

Teulière, C., Forestier, S., Lonini, L., Zhang, C., Zhao, Y., Shi, B., et al. (2015). Self-calibrating smooth pursuit through active efficient coding. *Rob. Auton. Syst.* 71, 3–12. doi: 10.1016/j.robot.2014.11.006

Tikhanoff, V., Cangelosi, A., Fitzpatrick, P., Metta, G., Natale, L., and Nori, F. (2008). "An open-source simulator for cognitive robotics research : the prototype of the iCub humanoid robot simulator," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems* (New York, NY), 57–61. doi: 10.1145/1774674.1774684

Vercher, J. L., Gauthiér, G. M., Guedon, O., Blouin, J., Cole, J., and Lamarre, Y. (1996). Self-moved target eye tracking in control and deafferented subjects: roles of arm motor command and proprioception in arm-eye coordination. *J. Neurophysiol.* 76, 1133–1144. doi: 10.1152/jn.1996.76.2.1133

Vercher, J. L., Volle, M., and Gauthier, G. M. (1993). Dynamic analysis of human visuo-oculo-manual coordination control in target tracking tasks. *Aviat. Sp. Environ. Med.* 64, 500–506.

Vicente, P., Jamone, L., and Bernardino, A. (2016). Online body schema adaptation based on internal mental simulation and multisensory feedback. *Front. Robot. AI* 3:7. doi: 10.3389/frobt.2016.00007

von Hofsten, C. (2004). An action perspective on motor development. *Trends Cogn. Sci.* 8, 266–272. doi: 10.1016/j.tics.2004.04.002

Wijesinghe, L. P., Antonelli, M., Triesch, J., and Shi, B. E. (2017). "Learning multisensory neural controllers for robot arm tracking," in *IEEE International Joint Conference on Neural Networks* (Anchorage, AK). 4150–4157. doi: 10.1109/IJCNN.2017.7966380

Zhang, C., Zhao, Y., Triesch, J., and Shi, B. E. (2014). "Intrinsically motivated learning of visual motion perception and smooth pursuit," in *IEEE International Conference on Robotics and Automation* (Hong Kong), 1902–1908. doi: 10.1109/ICRA.2014.6907110

Zhao, Y., Rothkopf, C., a., Triesch, J., and Shi, B. E. (2012). "A unified model of the joint development of disparity selectivity and vergence control," in *IEEE International Conference on Development and Learning and Epigenetic Robotics* (San Diego, CA), 1–6. doi: 10.1109/DevLrn.2012.6400876

Zhou, T., and Shi, B. E. (2016). Learning visuomotor transformations and end effector appearance by local visual consistency. *IEEE Trans. Cogn. Dev. Syst.* 8, 60–69. doi: 10.1109/TAMD.2015.2502758

Zmarz, P., and Keller, G. B. (2016). Mismatch receptive fields in mouse visual cortex. *Neuron* 92, 766–772. doi: 10.1016/j.neuron.2016.09.057

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH
published: 23 October 2018
doi: 10.3389/fnbot.2018.00068

Learning Inverse Statics Models Efficiently With Symmetry-Based Exploration

Rania Rayyes, Daniel Kubus and Jochen Steil*

Institut für Robotik und Prozessinformatik, Technische Universität Braunschweig, Braunschweig, Germany

Learning (inverse) kinematics and dynamics models of dexterous robots for the entire action or observation space is challenging and costly. Sampling the entire space is usually intractable in terms of time, tear, and wear. We propose an efficient approach to learn inverse statics models—primarily for gravity compensation—by exploring only a small part of the configuration space and exploiting the symmetry properties of the inverse statics mapping. In particular, there exist symmetric configurations that require the same absolute motor torques to be maintained. We show that those symmetric configurations can be discovered, the functional relations between them can be successfully learned and exploited to generate multiple training samples from one sampled configuration-torque pair. This strategy drastically reduces the number of samples required for learning inverse statics models. Moreover, we demonstrate that exploiting symmetries for learning inverse statics models is a generally applicable strategy for online and offline learning algorithms. We exemplify this by two different learning approaches. First, we modify the Direction Sampling approach for learning inverse statics models online, in a plain exploratory fashion, from scratch and without using a closed-loop controller. Second, we show that inverse statics mappings can be efficiently learned offline utilizing lattice sampling. Results for a 2R planar robot and a 3R simplified human arm demonstrate that their inverse statics mappings can be learned successfully for the entire configuration space. Furthermore, we demonstrate that the number of samples required for learning inverse statics mappings for 2R and 3R manipulators can be reduced at least by factors of approximately 8 and 16, respectively–depending on the number of discovered symmetries.

Keywords: symmetries, inverse statics models, inverse dynamics models, efficient learning, direction sampling, goal babbling

Edited by:
Vieri Giuliano Santucci, Istituto di scienze e tecnologie della cognizione (ISTC), Italy

Reviewed by:
Anja Philippsen, Center for Information and Neural Networks (CiNet), Japan
Tohid Alizadeh, Nazarbayev University, Kazakhstan

****Correspondence:***
Rania Rayyes
rrayyes@rob.cs.tu-bs.de

Received: *31 May 2018*
Accepted: *26 September 2018*
Published: *23 October 2018*

Citation:
Rayyes R, Kubus D and Steil J (2018) Learning Inverse Statics Models Efficiently With Symmetry-Based Exploration. Front. Neurorobot. 12:68. doi: 10.3389/fnbot.2018.00068

1. INTRODUCTION

The learning of motor capacities and skills has always been a core topic of the developmental approach to robot cognition (Asada et al., 2001), as mastering the body is fundamental for any embodied agent. Since the seminal work on human motor control in the 1990th (Wolpert and Kawato, 1998; Wolpert et al., 1998), it is widely believed that forward and inverse models play a crucial role in the motor control architectures. Numerous learning schemes have been proposed during the last decades for exploratory learning of robot forward and inverse kinematics, where in the developmental context exploratory learning without the initial constraint of a particular task or trajectory is the main focus. Note that in the latter case more specialized schemes can be applied both for kinematics (D'Souza et al., 2001) and dynamics (Peters and Schaal, 2008; Meier et al., 2016) and the learning problem is locally convex which simplifies the task significantly. Motor control for the entire configuration space, however, remains a major challenge because sampling the entire action or observation space is usually very costly and the non-convexity of the model (e.g., due to kinematic redundancy) poses additional problems.

Efficiency is one of the major challenges in learning (inverse) kinematics and dynamics models. Reducing the number of required samples to learn these models in practical experiments is beneficial regarding time and hardware costs. We therefore propose *symmetry-based exploration* to effectively reduce the number of required samples. This can be done by exploiting the mapping properties to learn a model that is valid for the entire action/observation space. For example, it is a particular property of inverse statics maps (ISMs) (i.e., the map that assigns a required static torque to maintain a desired joint configuration of the robot) that multiple configurations require the same absolute static torque to be maintained. We denote this configuration set as symmetry set. We exploit the functional relation between configurations in the symmetry set to show that learning ISMs can be done very efficiently by exploring only one configuration and learning the corresponding symmetric configurations. To this aim, we propose a scheme to discover and learn symmetries, and then we exploit these symmetries to drastically reduce the number of required samples regardless of the particular learning scheme. The paper demonstrates the generic nature of the symmetry concept to accelerate the learning process through exploiting symmetries with different learning schemes online and offline.

Learning ISMs has previously been done offline only and by using a feedback-controller to collect samples and to enhance an already existing model (e.g., Luca and Panzieri, 1993; Xie et al., 2008). In this paper, Direction Sampling (Rolf, 2013), which has been previously proposed as an extension of Goal Babbling (Rolf et al., 2011) to learn inverse kinematics (IK), is modified to learn ISMs also online, from scratch and without using any controller in a plain exploratory fashion. Learning ISMs in an exploratory fashion is challenging as the straightforward application of random torques bears the risk to destroy any manipulator if no further safety layers are present and to respect joint-wise torque limits alone does not solve this problem, other than in kinematics, where joints limits can be enforced easily and without endangering the robot hardware. Hence, the exploration may yield inadmissible torques which result in accelerating the robot manipulator and the robot hitting its joint limits[1]. Consequently, the learner will be disturbed because of the resulting invalid training sample consisting of inadmissible torque which is not corresponding to the joint limits' configuration where the robot settles in. To avoid this situation, torque combination limits must be considered in addition to the joint-wise torque limits. We therefore explore and learn the set of admissible static torques to overcome this issue as explained in detail in section 5.1.

These aforementioned challenges also illustrate more restrictions and difficulties of learning ISMs in comparison to learning IK. For example, the application of a torque produces dynamics, other than in the kinematics domain where application of a joint command can be treated as instantaneously effective, because the underlying joint controllers hide and control the dynamics. Furthermore, the training samples in IK are always valid samples since the end-effector pose always corresponds to a valid robot configuration even when the robot hits its joint limits, which is not the case in ISMs. Moreover, IK usually maps from Cartesian (observation) space to configuration (action) space, i.e., from a lower dimensional space to a higher dimensional one, while the dimensions of observation and action spaces in ISMs are usually identical since ISMs map from configuration (observation) space to motor (action) space. Learning the mapping between spaces with identical dimensions is more difficult as both dimensions scale with the number of DoFs. Consequently, more samples are required to learn the model in contrast to IK. Hence, exploiting symmetries and exploring only a small part of the configuration space is also motivated to mitigate the curse of dimensionality problem. It reduces the number of required samples as the efficiency factor increases for higher DoFs. For instance, it increases to 8 for a 2R planar manipulator and to 16 for a 3R robot manipulator as illustrated in section 7.

The remainder of the paper is structured as follows: Section 2 reviews related work. Section 3 introduces the concept of symmetries. Section 4 explains symmetry discovery and symmetry exploitation in learning. Section 5 addresses learning ISMs online and explains the proposed Constrained Direction Sampling. Lattice sampling is introduced briefly in section 6. Section 7 presents experimental results and the efficiency gained by exploiting symmetries for learning ISMs which is illustrated by Constrained Direction Sampling (online) and a batch learning technique using lattice sampling for a 2R and a 3R manipulators. Section 8 concludes the work.

2. RELATED WORK

Our main goal is increasing the efficiency of learning models, in particular for learning inverse statics. As learning ISMs has been done previously only offline, we modified the Direction Sampling method (Rolf, 2013) for learning ISMs online as well. This paper therefore discusses three major points: learning efficiently, learning inverse statics models, and online goal-directed approaches. This section presents the previous related work.

2.1. Learning Efficiently

Various approaches have previously been proposed for tackling the efficiency problem of learning. Some previous research proposed exploring the observation space instead of the action space to avoid the curse of dimensionality. For instance, learning IK by exploring the observation space (Cartesian space) and learning only one configuration for each pose to mimic infants

[1]Obviously, in practical applications, a software joint limit is employed to avoid reaching the hardware joint limit.

efficient sensorimotor learning (e.g., Rolf et al., 2011; Rolf and Steil, 2014; Rayyes and Steil, 2016) instead of learning forward kinematics mappings by exploring the higher dimensional action space (configuration space) e.g., Motor Babbling (Demiris and Meltzoff, 2008).

Other research proposed that online learning of inverse models can be done in part of the workspace only in order to increase the efficiency and reduce the number of required samples (Rolf et al., 2011; Baranes and Oudeyer, 2013), since online learning approaches have the tendency to require more samples than offline methods. Efficient exploration by efficient sampling (active policy iteration) was proposed in Akiyama et al. (2010), however it has been proposed for batch learning only. Efficient learning has been also addressed for solving different tasks (e.g., Şimşek and Barto, 2006) based on Markov Decision Process and reward function. In this paper, we propose *symmetry-based exploration* to learn ISMs for the entire configuration space effectively by exploring a small part of it and exploiting the symmetries of ISMs which reduces the number of required samples. The proposed strategy is applicable for online and offline learning schemes.

2.2. Learning Inverse Statics Models

Compensating forces and torques due to gravity is very important for advanced model-based robot control. The gravitational terms of the inverse dynamics models are usually computed either by estimating inertial parameters of the links or from CAD data of the robot. However, if no appropriate model exists e.g., for advanced complex robots or for soft robots, or if no prior knowledge on the inertial parameters of the links is available, *learning* these gravitational terms is a promising option. Previous research on learning ISMs has been done offline using a closed-loop controller to collect training data and often to enhanced existing (parametric) models (e.g., Luca and Panzieri, 1993; Xie et al., 2008). Early data-driven gravity compensation approaches are based on iterative procedures for end-point regulation (De Luca and Panzieri, 1994; De Luca and Panzieri, 1996). Recent works (Giorelli et al., 2015; Thuruthel et al., 2016b) have proposed data-driven learning techniques to control the end-point of continuum robots in task space. Where ISMs map between the desired end effector poses and the cable tensions. However, feedback controllers and inefficient Motor Babbling were implemented to obtain the training data and to learn ISMs offline only. In contrast, we propose learning ISMs online, in an exploratory fashion, from scratch and without using a closed-loop controller. Besides, we exploit the symmetry properties of ISMs to learn ISMs efficiently online and offline for the entire configuration space.

2.3. Goal Babbling and Direction Sampling

Various schemes have been proposed to replicate human movement skill learning and human motor control based on internal models (Wolpert et al., 1998), i.e., learning forward models (e.g., Motor Babbling Demiris and Meltzoff, 2008), and inverse models (e.g., distal teachers Jordan and Rumelhart, 1992 and feedback error learning; Gomi and Kawato, 1993). In contrast to Motor Babbling where the robot executes random motor commands and the outcomes are observed, there is evidence that even infants do not behave randomly but rather demonstrate goal-directed motion already few days after birth (von Hofsten, 1982). They learn how to reach by trying to reach and they iterate their trails to adapt their motion. Hence, Goal Babbling was proposed and inspired by infant motor learning skills for direct learning of IK within a few 100 samples (Rolf et al., 2010, 2011). Various other schemes were proposed for learning IK e.g., direct learning of IK (D'Souza et al., 2001; Thuruthel et al., 2016a) and incremental learning of IK (Vijayakumar et al., 2005; Baranes and Oudeyer, 2013).

To apply Goal Babbling, a set of predefined targets, e.g., a set of positions to be reached, is required and then used to obtain the IK which is valid only in the predefined area. Direction Sampling (Rolf, 2013) has been proposed as an extension of Goal Babbling, to overcome the need for predefined targets and gradually discover the entire workspace. The targets are generated while exploring and the IK is learned simultaneously. In previous work, we already illustrated the scalability of online Goal Babbling with Direction Sampling in higher dimensional sensorimotor spaces up to 9-DoF COMAN floating-base (Rayyes and Steil, 2016). Goal Babbling has also been extended to learn IK in restricted areas (Loviken and Hemion, 2017) and to other domains e.g., speech production (Moulin-Frier et al., 2013; Philippsen et al., 2016) and tool usage (Forestier and Oudeyer, 2016). Besides, it has been also applied to soft robots (Rolf and Steil, 2014). However, it is striking that none of these schemes have been extended or transferred to learn the forward or inverse dynamics. As Goal Babbling shows high scalability and adaptability in "learning while behaving" fashion, we focus in this paper on learning ISMs, as a first step in the direction of exploratory dynamics leaning, by modifying the previously proposed Direction Sampling based on online Goal Babbling.

3. INVERSE STATIC MODELS AND SYMMETRIC CONFIGURATIONS

In this section, we first explore fundamental properties of ISMs, subsequently devise the concept of symmetries and then define the notion of primary and secondary symmetric configurations which are finally illustrated with a 2R planar manipulator. We will use the term torques instead of generalized actuator forces as our main target are manipulators with revolute joints only.

3.1. Properties of Inverse Statics Maps

ISMs map from configuration space, which constitutes the observation space, to motor space, which represents the action space. The dimensionality of the domain and codomain in ISMs are therefore identical. ISMs are many-to-one mappings, i.e., multiple configurations require the same torque to be maintained as illustrated in **Figure 1**.

We aim to learn the map $\boldsymbol{G}$ which assigns to each joint configuration $\boldsymbol{q} \in \mathcal{Q}_p$ a torque $\boldsymbol{\tau} \in \mathcal{T}_s$ required to maintain this configuration:

$$\boldsymbol{G} : \mathcal{Q}_p \rightarrow \mathcal{T}_s, \qquad \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{1}$$

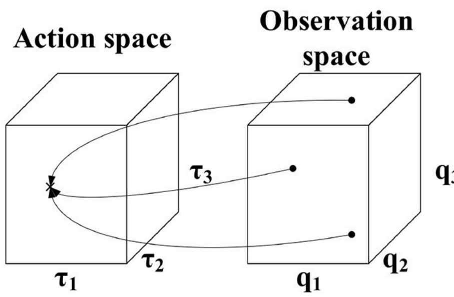


FIGURE 1 | Characteristics of ISMs. The same torque is required to maintain different configurations.

$\mathcal{Q}_p$ is the set of permissible configurations while $\mathcal{T}_s$ is the set of required static torques to maintain these configurations. $\boldsymbol{G}$ typically associates each member of the set $\mathcal{T}_s$ with more than one member of the domain $\mathcal{Q}_p$. There typically exist respective level sets

$$\mathcal{L}_{\boldsymbol{\tau}} = \{\boldsymbol{q} : \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau}\} \tag{2}$$

with cardinalities $|\mathcal{L}_{\boldsymbol{\tau}}| > 1$ for admissible torque vectors $\boldsymbol{\tau} \in \mathcal{T}_s$.

3.2. Symmetric Configurations

We define the concept of symmetries as following:

Consider two level sets $\mathcal{L}_{\boldsymbol{\tau}_i}$ and $\mathcal{L}_{\boldsymbol{\tau}_j}$ where

$$\boldsymbol{\tau}_i = \boldsymbol{\Upsilon}\boldsymbol{\tau}_j, \boldsymbol{\Upsilon} = \operatorname{diag}(\delta_1, \dots, \delta_n), \qquad \delta_k = \pm 1, \quad \sum_{k=1}^{n} \delta_k < n \tag{3}$$

i.e., the elements in $\boldsymbol{\tau}_i$ and $\boldsymbol{\tau}_j$ differ w.r.t. their sign. Here, n denotes the number of DoFs and $\operatorname{diag}(\delta_1, \dots, \delta_n)$ denotes a diagonal matrix with $\delta_1, \dots, \delta_n$ on its main diagonal. We define $\breve{\mathcal{L}}_{\boldsymbol{\tau}}$ as

$$\breve{\mathcal{L}}_{\boldsymbol{\tau}} = \bigcup_{k=1}^{2^n} \mathcal{L}_{\boldsymbol{\tau}_k} \tag{4}$$

$\breve{\mathcal{L}}_{\boldsymbol{\tau}}$ is the union of all level sets fulfilling Equation (3), i.e., the union of the level sets which have the same absolute value of the elements in the torque vector.

Two classes of configurations in these level sets can be distinguished. Primary symmetric configurations, also denoted as primary symmetries, constitute those pairs of configurations $\boldsymbol{q}_r, \boldsymbol{q}_s \in \breve{\mathcal{L}}_{\boldsymbol{\tau}}$ for which

$$\boldsymbol{M}_{r,s}\boldsymbol{q}_r + \boldsymbol{N}_{r,s}\boldsymbol{q}_s = \boldsymbol{d}_{r,s} \tag{5}$$

holds – where $\boldsymbol{d}_{r,s} \in \mathbb{R}^n$ and $\boldsymbol{M}_{r,s}, \boldsymbol{N}_{r,s} \in \mathbb{R}^{n\times n}$ are constant (in particular independent of the choice of $\boldsymbol{\tau}$). The set of all configurations in $\breve{\mathcal{L}}_{\boldsymbol{\tau}}$ which are directly or transitively related by Equation (5) is called the set of primary symmetries (SPS) denoted by $\mathcal{S} \subset \breve{\mathcal{L}}_{\boldsymbol{\tau}}$.

Secondary symmetric configurations, also denoted as secondary symmetries, constitute those configurations in $\breve{\mathcal{L}}_{\boldsymbol{\tau}}$ for which at least one of $\boldsymbol{d}_{r,s}, \boldsymbol{M}_{r,s}, \boldsymbol{N}_{r,s}$ is a function of $\boldsymbol{q}$ and/or $\boldsymbol{\tau}$.

3.2.1. Symmetric Configurations of a Planar 2R Manipulator

To exemplify the idea of primary symmetries and secondary symmetries, **Figure 2A** shows all symmetric configurations of a 2R planar robot. There are 16 configurations which need the same absolute static torque to be maintained and they can be separated into two disjoint sets $\mathcal{S}_A$ (blue) and $\mathcal{S}_B$ (red) of 8 configurations each.

The set $\mathcal{S}_A$ constitutes a set of primary symmetries. The symmetric configurations in $\mathcal{S}_A$ are also geometrically symmetric as illustrated in **Figure 2A**, it is therefore, easy to find the functional relation between them with the linear equation given in Equation (5). Similarly, the set $\mathcal{S}_B$ constitutes a set of primary symmetries as well. These two sets are secondary symmetric to each other as $\mathcal{S}_A$ and $\mathcal{S}_B$ have identical absolute static torques. The secondary symmetries occur by relating configurations from $\mathcal{S}_A$ with those from $\mathcal{S}_B$, however; there is no simple closed form functional relations between these two sets. We will therefore consider only primary symmetries in our experimental results.

For visualization purposes, we use component-wise level sets for the 2R planar manipulator (cf. **Figure 2A**) as defined below and illustrated in **Figure 2B**:

$$\begin{aligned} \mathcal{L}_{\tau_1} &= \left\{\boldsymbol{q} : \boldsymbol{G}(\boldsymbol{q}) = [\tau_1, \tau_2]^T, \tau_2 \in \mathbb{R}\right\}, \\ \mathcal{L}_{-\tau_1} &= \left\{\boldsymbol{q} : \boldsymbol{G}(\boldsymbol{q}) = [-\tau_1, \tau_2]^T, \tau_2 \in \mathbb{R}\right\} \end{aligned} \tag{6}$$

$$\begin{aligned} \mathcal{L}_{\tau_2} &= \left\{\boldsymbol{q} : \boldsymbol{G}(\boldsymbol{q}) = [\tau_1, \tau_2]^T, \tau_1 \in \mathbb{R}\right\}, \\ \mathcal{L}_{-\tau_2} &= \left\{\boldsymbol{q} : \boldsymbol{G}(\boldsymbol{q}) = [\tau_1, -\tau_2]^T, \tau_1 \in \mathbb{R}\right\} \end{aligned} \tag{7}$$

$\mathcal{L}_{\pm\tau_1}$ and $\mathcal{L}_{\pm\tau_2}$ fix one component of $\boldsymbol{\tau}$ while the other one is not restricted. All pairwise intersection points of component-wise level sets $\mathcal{L}_{\pm\tau_1}$ and $\mathcal{L}_{\pm\tau_2}$ constitute symmetric configurations as they have the same absolute values of the elements in the torque vectors and hence fulfill Equations (2, 3).

Note that the component-wise level set is different from the level set which is defined in Equation (2). The component-wise level set fixes only one component of $\boldsymbol{\tau}$, while the level set in Equation (2) fixes all components of $\boldsymbol{\tau}$. Based on Equations (2–4), the level sets for the 2R robot illustrated in **Figure 2A** are:

$$\left.\begin{aligned} \breve{\mathcal{L}}_{\boldsymbol{\tau}} &= \bigcup_{k=1}^{2^2} \mathcal{L}_{\boldsymbol{\tau}_k} \\ \mathcal{L}_{\boldsymbol{\tau}_1} &= \left\{\boldsymbol{q} : \boldsymbol{G}(\boldsymbol{q}) = [+\tau_1, +\tau_2]^T\right\} \\ \mathcal{L}_{\boldsymbol{\tau}_2} &= \left\{\boldsymbol{q} : \boldsymbol{G}(\boldsymbol{q}) = [+\tau_1, -\tau_2]^T\right\} \\ \mathcal{L}_{\boldsymbol{\tau}_3} &= \left\{\boldsymbol{q} : \boldsymbol{G}(\boldsymbol{q}) = [-\tau_1, +\tau_2]^T\right\} \\ \mathcal{L}_{\boldsymbol{\tau}_4} &= \left\{\boldsymbol{q} : \boldsymbol{G}(\boldsymbol{q}) = [-\tau_1, -\tau_2]^T\right\} \end{aligned}\right\} \tag{8}$$

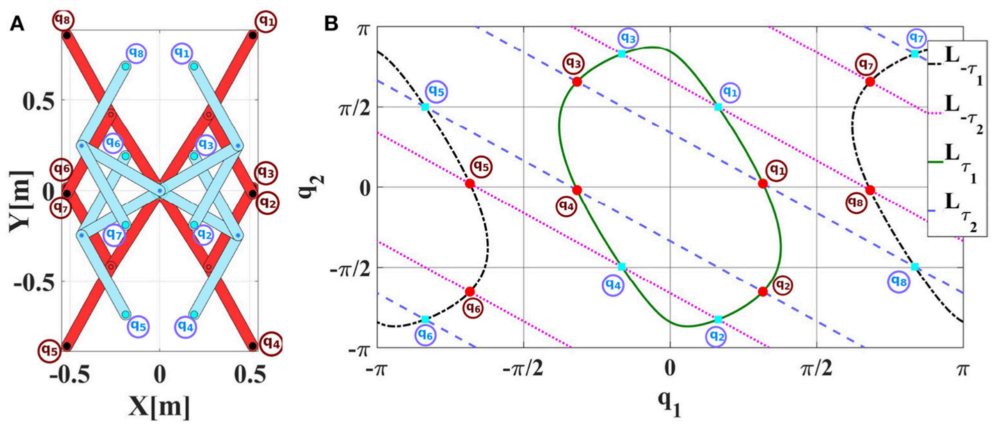


FIGURE 2 | (A) Symmetric configurations of a 2R planar robot which require the same absolute static torque to be maintained. Configuration pairs in each configuration set illustrated in blue $\mathcal{S}_A$ (and red $\mathcal{S}_B$, respectively) are primary symmetric to each other in the same set. The two sets are secondary symmetric to each other. Note that the manipulator is stretched out to the right in its zero configuration and that the gravity vector points downwards into negative y-direction.
(B) Component-wise level sets $\mathcal{L}_{\tau_1}$,$\mathcal{L}_{\tau_2}$,$\mathcal{L}_{-\tau_1}$,$\mathcal{L}_{-\tau_2}$ of the 2R planar manipulator. The 16 intersection points constitute symmetric configurations. Their colors and numbers correspond to the configurations shown in **Figure 2A**. The numbers are based on Equation (11).

Each level set comprises 4 configurations corresponding to 4 points in the pairwise intersections of the component-wise level sets in **Figure 2B**. Therefore, the symmetric configurations form the union of the level sets $\check{\mathcal{L}}_{\boldsymbol{\tau}}$ and the pairwise intersections of component-wise level sets $\bigcap_{i=1}^{2^2} \mathcal{L}_{\tau_i}$.

Like the configurations in **Figure 2A**, the 16 intersection points in **Figure 2B** can be separated into the two disjoint sets $\mathcal{S}_A$ and $\mathcal{S}_B$ indicated by the color of the points. The numbers indicate the corresponding torque (intersection point) for each configuration in **Figure 2A** which fulfill Equation (11) as well. We can also derive the required torque for each joint geometrically from **Figure 2A** and relate it with **Figure 2B**. Following the right-hand rule, we can detect the sign of the torque for each joint. In this setup, the zero configuration is where the arm stretched out to the right. Every torque of a joint whose link is located on the right side of a virtual vertical line/plane will have a positive sign. For instance, for $\boldsymbol{q_1}$ in $\mathcal{S}_B$ (red), we can imagine a vertical line passing through the origin and a second vertical line passing through the second joint axis. Both links are on the right side of the lines so their torques are positive. On the contrary, both links of $\boldsymbol{q_8}$ in $\mathcal{S}_B$ (red) are on the left side of the imaginary vertical lines. So their torques are negative.

4. ACCELERATING LEARNING BY EXPLOITING SYMMETRIES

Each torque vector $\boldsymbol{\tau}$ with identical absolute values of its elements corresponds to a non-singleton set $\check{\mathcal{L}}_{\boldsymbol{\tau}}$ of configurations. Hence, functional relations between the configurations in $\check{\mathcal{L}}_{\boldsymbol{\tau}}$ can be exploited to generate training data and associate each configuration in $\check{\mathcal{L}}_{\boldsymbol{\tau}}$ with its applied torque vector $\boldsymbol{\Upsilon}'\boldsymbol{\tau}$ by observing just one configuration from $\check{\mathcal{L}}_{\boldsymbol{\tau}}$ where

$$\boldsymbol{\Upsilon}' = \mathrm{diag}\,(\delta_1, \ldots, \delta_n) \tag{9}$$

$$\delta_k = \pm 1, \quad \sum_{k=1}^{n} \delta_k \leq n$$

Before symmetric configurations can be exploited in this way, they need to be discovered and the functional relations between them need to be learned or inferred. Symmetric configurations can be discovered by applying suitable torque profiles to the manipulator (cf. section 4.1). Once a number of n_{sym} functional relations is determined, each applied motor command $\boldsymbol{\tau}_i$ generates a sample $(\boldsymbol{q}_i, \boldsymbol{\tau}_i)$ as well as $n_{sym} - 1$ further samples $(\boldsymbol{q}_j, \boldsymbol{\Upsilon}'_i\boldsymbol{\tau}_i), i \neq j$ obtained by evaluating the previously established functional relations between symmetric configurations which are explained in section 4.2. Increasing the efficiency by exploiting symmetries and limiting the exploration to only one part of configuration space is explained in section 4.3.

4.1. Discovering Symmetric Configurations

For symmetry discovery, sequences of suitable torque profiles are applied with the same absolute starting and ending torque values. **Algorithm 1** shows the required steps for discovering the symmetries associated with a single torque vector $\boldsymbol{\tau}^*$.

Let $\boldsymbol{\tau}^{pr}$ denote a torque profile. Starting from the home configuration $\boldsymbol{q}^{home}$, a number n_{pr} of torque profiles $\boldsymbol{\tau}_i^{pr}$ are generated using splines (cf. **Figure 3**) and applied sequentially, where $\boldsymbol{\tau}_i^{pr}$ is the ith torque profile. Each torque profile has $k = 1, .., n_{s_i}$ time steps. These torques profiles are applied with start and end-point constraints on their derivatives, i.e., $\dot{\boldsymbol{\tau}}_i^{pr}[1] = \dot{\boldsymbol{\tau}}_i^{pr}[n_{s_i}] = \boldsymbol{0}$, which is required to obtain a smooth trajectory.

For each torque profile $\boldsymbol{\tau}_i^{pr}$, $\boldsymbol{\tau}_i^{pr}[1] = \boldsymbol{\Upsilon}'\boldsymbol{\tau}^* \wedge \boldsymbol{\tau}_i^{pr}[n_{s_i}] = \boldsymbol{\Upsilon}'\boldsymbol{\tau}^*$ holds. Probability distributions p_n and p_τ are utilized to draw n_{s_i} samples and to generate intermediate torques in each profile, respectively.

After successful application of a torque profile, $\boldsymbol{\tau}_i^{p}[n_{s_i}]$ is applied as long as the manipulator has not settled yet, i.e., $\boldsymbol{\tau}_i^{p}[n_{s_i}]$

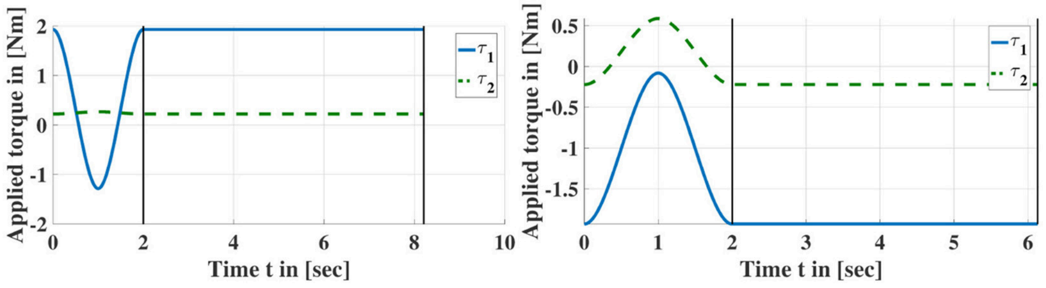


FIGURE 3 | Examples of torque profiles for symmetry discovery. First, a torque spline is applied with the same initial and terminal absolute torque values. Subsequently, a constant torque is commanded until the manipulator settles at a configuration.

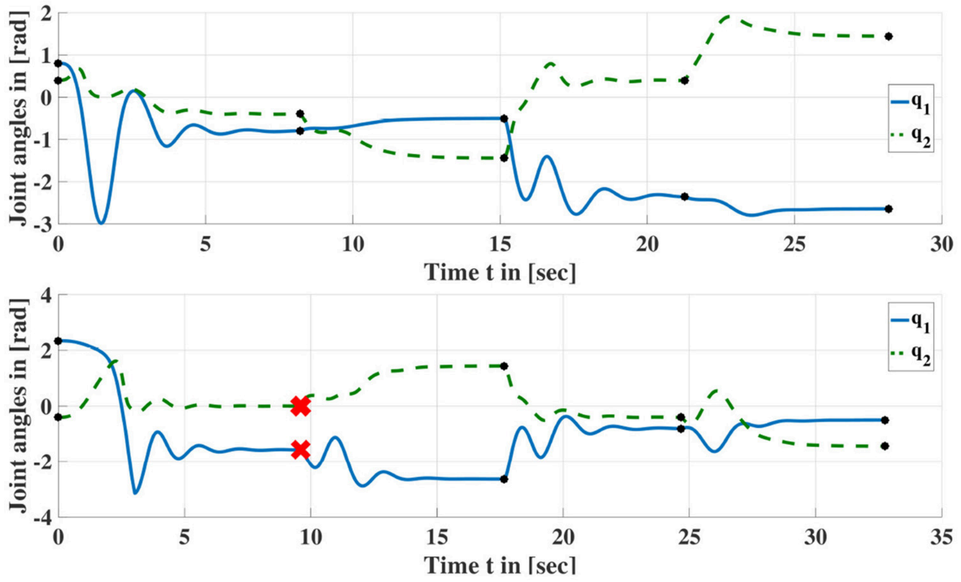


FIGURE 4 | Joint trajectories resulting from applying sequences of torque profiles according to **Figure 3**. Red crosses indicate that the joint limits have been reached and the manipulator returned to its home configuration. Black dots indicate the end of a profile where $\boldsymbol{\tau}_i^{pr}[n_{S_i}]$ is applied until the manipulator has settled down in this configuration i.e., the manipulator has stopped moving. The corresponding configurations are entered into $\check{\mathcal{L}}'_\tau$ (cf. **Algorithm 1)**.

is applied until the manipulator stops moving. By reverting to the same torque magnitude at the end of each profile but applying different intermediate torques, a primary or secondary symmetric configuration can be reached. If the manipulator settles in a valid configuration, this configuration $\boldsymbol{q}$ is recorded and added to the discovered set $\check{\mathcal{L}}'_\tau$ (if is not already contained in it) associated with the torque $\boldsymbol{\Upsilon}'\boldsymbol{\tau}^*$ and the sequence is continued with the next profile. If the manipulator reaches its joint limits during or after application of a torque profile, it goes back to its home configuration $\boldsymbol{q}^{home}$ and the sequence is continued with the next profile. The discovered symmetries are marked as primary symmetries if they can be related according to Equation (5).

Figure 3 shows exemplary torque profiles and **Figure 4** shows two joint trajectories resulting from the application of such torque profiles. 5 and 4 symmetric configurations are discovered, respectively including the initial configurations. Note that n_{pr} depends on the geometrical structure and the number of joints of the robot.

4.2. The Functional Relations Between Symmetric Configurations

The functional relations between the primary symmetries according to Equation (5) can be determined by established multiple linear regression techniques (cf. e.g., Draper and Smith, 1998). These learned relations can then be utilized to compute the symmetric configurations for each observed $\boldsymbol{q}$ with the corresponding $\boldsymbol{\tau}$ required to maintain it.

When some geometrical information about the manipulator is available and when the primary symmetries are also geometrically symmetric to each other, then the functional relations between them are easily inferred utilizing the functional relations of geometrical symmetries.

For example, the functional relations between primary symmetries for the 2R planar robot illustrated in

Algorithm 1 Symmetry Discovery using Torque Profiles

```
1: function DISCOVERSYM($\boldsymbol{\tau}^*, \boldsymbol{q}_{min}, \boldsymbol{q}_{max}, n_{pr}, p_n, p_\tau$)
2:     $\check{\mathcal{L}}'_\tau = \emptyset$
3:     for $i = 1, \ldots, n_{pr}$
4:         $settled = false$
5:         $n_{s_i}$ = sample probability distribution $p_n$
6:         $\boldsymbol{\tau}_i^{pr}$ = generate Torque Profile ($\boldsymbol{\tau}^*, p_\tau, n_{s_i}$)
7:         for $j = 1, \ldots, n_{s_i}$
8:             apply torque $\boldsymbol{\tau}_i^{pr}[j]$
9:             observe current configuration $\boldsymbol{q}$
10:            if $\boldsymbol{q}$ exceeds joint limits
11:                go home
12:                $settled = true$
13:                break
14:            end if
15:        end for
16:        while$\neg settled$
17:            apply torque $\boldsymbol{\tau}_i^{pr}[n_{s_i}]$
18:            observe current configuration $\boldsymbol{q}$
19:            if $\boldsymbol{q}$ exceeds joint limits
20:                go home
21:                $settled = true$
22:            elseif robot has settled in $\boldsymbol{q}$
23:                add $\boldsymbol{q}$ to $\check{\mathcal{L}}'_\tau$
24:                $settled = true$
25:            end if
26:        end while
27:    end for
28:    return $\check{\mathcal{L}}'_\tau$
29: end function
```

Figure 2A are given in Equations (10, 11) according to Equation (5) and applying elementary geometric considerations:

$$\mathcal{S} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, \boldsymbol{q}_4, \boldsymbol{q}_5, \boldsymbol{q}_6, \boldsymbol{q}_7, \boldsymbol{q}_8\} \tag{10}$$

$$\left.\begin{aligned}
\boldsymbol{q}_1 &= [q_1, q_2]\\
\boldsymbol{q}_2 &= [q_1, -q^* - q_1]\\
\boldsymbol{q}_3 &= [-q_1, q^* + q_1]\\
\boldsymbol{q}_4 &= [-q_1, -q^* + q_1]\\
\boldsymbol{q}_5 &= [q_1 + \pi, q^* - q_1]\\
\boldsymbol{q}_6 &= [q_1 + \pi, -q^* - q_1]\\
\boldsymbol{q}_7 &= [-q_1 - \pi, q^* + q_1]\\
\boldsymbol{q}_8 &= [-q_1 - \pi, -q^* + q_1]\\
q^* &= q_1 + q_2
\end{aligned}\right\} \tag{11}$$

$\mathcal{S}$ is the set of primary symmetries, $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots \boldsymbol{q}_8\}$ are the symmetric robot configurations, q_1, q_2 are the robot joint angles and q^* is a virtual joint angle illustrated in **Figure 5**.

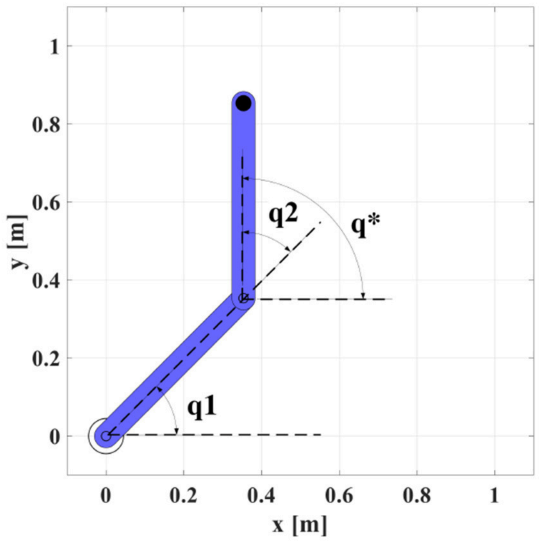


FIGURE 5 | The physical and virtual joint angles of a 2R manipulator to calculate the set of primary symmetries $\mathcal{S}$.

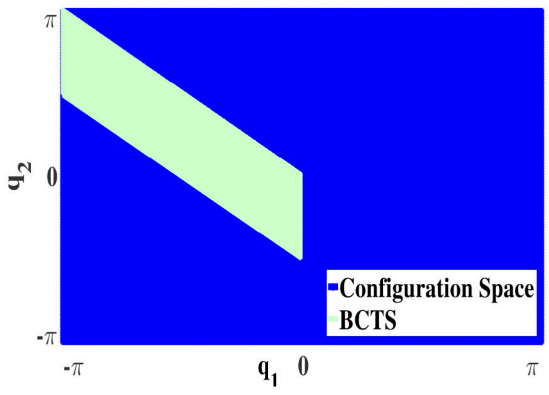


FIGURE 6 | BCTS in the configuration space for a 2R planar manipulator.

4.3. Increasing Efficiency by Exploiting Symmetries

4.3.1. Bijective Configuration-Torque Set (BCTS)

Owing to the symmetry properties of ISMs, only a fraction of the configuration space needs to be explored. We denote this subspace as bijective configuration-torque set (**BCTS**). The **BCTS** is a set of configurations which contains exactly one unique configuration $\boldsymbol{q}$ for each admissible absolute static torque $\boldsymbol{\tau}$. **BCTS** is determined based on the set of primary symmetries. For example, **Figure 6** illustrates the **BCTS** (green area) for the 2R planar robot (cf. **Figure 2A**) which is determined based on the set of primary symmetries $\mathcal{S}$ given in Equations (10, 11).

As configurations outside the **BCTS** are symmetric to those inside the **BCTS**, ISMs can be learned for the entire configuration space by exploring merely the **BCTS** and exploiting the functional relations between symmetries. Constraining the exploration to discover the **BCTS** only increases the efficiency of learning and decreases the number of required samples to learn ISMs as we explore non-symmetric samples only.

For the 2R planar robot shown in **Figure 2A**, the currently achievable reduction factor r w.r.t. required samples is $r = 8$ as the primary symmetry set has cardinality 8, while exploiting secondary symmetries would further increase r up to 16. For the 3R simplified human arm (Babiarz et al., 2015) illustrated in **Figure 9**, the cardinality of the primary symmetry set increases to $r = 16$. Exploiting secondary symmetries would again yield far higher reduction factors depending on the properties of the manipulator, however, we currently have no means to exploit them.

5. LEARNING INVERSE STATIC MODELS ONLINE

In order to learn ISMs for the entire configuration space online, from scratch, in a plain exploratory fashion and without using a feedback controller , we employ Direction Sampling (Rolf, 2013). However, to apply it successfully for bootstrapping ISMs, several modifications to the original scheme are necessary. We therefore propose Constrained Direction Sampling. First, the constraint in form of the set of statically admissible torques is introduced.

5.1. Set of Static Torques (SST)

In the established Goal Babbling and Direction Sampling (Rolf et al., 2011; Rolf, 2013; Rayyes and Steil, 2016), exploratory noise is added in the action space in order to explore and learn new configurations. However, adding this exploratory noise to motor commands (torques) in ISMs may yield inadmissible torques. Consequently, the robot will accelerate and hit its joint limits which results in invalid training samples (inadmissible torques which don't correspond to the joint limits' configuration where the robot settles in).

In order to avoid such situations, the set of statically admissible torques (**SST**) should be estimated beforehand or learned and the exploration should be constrained to the **SST**. Therefore, we modify Goal Babbling and Direction Sampling in this paper to limit the exploration to this set with applying the nearest neighbor strategy. These modified approaches are termed Constrained Goal Babbling and Constrained Direction Sampling, respectively.

The set of statically admissible torques (**SST**) is defined as:

$$\mathcal{T}_s = \left\{ \boldsymbol{\tau} \mid \exists \boldsymbol{q} \in \mathcal{Q}_p : \boldsymbol{\tau} - \boldsymbol{G}(\boldsymbol{q}) = 0 \right\} \tag{12}$$

Each time the robot hits its joint limits during the learning process, the corresponding torque is marked as inadmissible and the **SST** estimate is updated accordingly. Delaunay triangulation is used to estimate the **SST** boundary. Exploratory noise (cf. Equation 14) will be added to the static torque and the nearest neighbor algorithm is employed to assign each invalid torque to a valid one before execution. **Figure 7A** shows the **SST** (blue points) for a 2R planar manipulator with specific joint limits and illustrates that applying the original Goal Babbling and adding explanatory noise might result in torques outside the **SST** i.e., inadmissible torques. After applying Constrained Goal Babbling, the exploration is limited to the **SST** as illustrated in **Figure 7B**; this avoids generating invalid training samples and avoids the robot hitting its joints limits as well. To save time, this exploration can be performed in conjunction with symmetry discovery as detailed in section 5.4.

5.2. Constrained Direction Sampling for Learning ISMs

Originally, Direction Sampling was proposed in Rolf (2013) to learn IK. In this paper, we modify Direction Sampling to learn ISMs by incorporating **SST** constraints and the nearest neighbor strategy. Moreover, our approach can be applied to robots with both prismatic and revolute joints. **Algorithm 2** shows the individual steps of the Constrained Direction Sampling. The initial inverse estimate $\hat{\boldsymbol{G}}(\boldsymbol{q})$ at time instant $t = 0$ yields some constant default torque $\hat{\boldsymbol{G}}(\boldsymbol{q}) = \boldsymbol{\tau}^{home}$ corresponding to some comfortable default configuration (home posture) $\boldsymbol{q}^{home}$ (cf. line 2 in **Algorithm 2**). The robot starts exploring from its home posture $\boldsymbol{q}^{home}$ and the targets are generated along a random direction $\boldsymbol{\Delta q}$ as given in Equation (13):

$$\boldsymbol{q}^*_t = \boldsymbol{q}^*_{t-1} + \frac{\varepsilon}{\|\boldsymbol{w}^T \boldsymbol{\Delta q}\|} \cdot \boldsymbol{\Delta q} \tag{13}$$

where $\boldsymbol{q}^*_t$ is the currently generated target, $\boldsymbol{q}^*_{t-1}$ is the previous one, $\boldsymbol{w}$ is a weighting vector as the joint space may be noncommensurate if both prismatic and revolute joints occur (here $\boldsymbol{w} = \boldsymbol{1}$ as we consider revolute joints only), ε is the step-width between the generated targets, and t indicates the time-step. $\boldsymbol{q}^{home}$ is selected as a target with some probability $p^{home} \ll 1$. The agent tries to reach and maintain each generated target $\boldsymbol{q}^*_t$ using the online Goal Babbling basic scheme (GBSCHEME, cf. **Algorithm 2**) as following: The current inverse estimate for each generated target $\boldsymbol{q}^*_t$ represents the motor torque $\hat{\boldsymbol{\tau}}^*_t$ required to maintain this target. Correlated exploratory noise $\boldsymbol{\sigma}$ (Rolf et al., 2011) is added to discover and learn new configurations as specified in Equation (14) (cf. line 15 in **Algorithm 2**):

$$\boldsymbol{\tau}^+_t = \hat{\boldsymbol{\tau}}^*_t + \boldsymbol{\sigma}(\boldsymbol{q}^*_t, t) \tag{14}$$

$\boldsymbol{\tau}^+_t$ is the torque which is applied to the robot if $\boldsymbol{\tau}^+_t \in \mathcal{T}_s$ holds or (if $\boldsymbol{\tau}^+_t \notin \mathcal{T}_s$) it will be assigned to the nearest valid one (cf. line 16 in **Algorithm2**), the outcome ($\boldsymbol{q}^+_t$) is then observed (cf. line 19 in **Algorithm 2**) and the inverse estimate is updated immediately (cf. line 21 in **Algorithm 2**). In simulation, a full dynamic simulation based on the forward dynamics model (Craig, 1986) of the robot is required.

The robot tries to explore along the desired direction until its actual direction of motion deviates from the intended one more than φ degrees. For $\varphi = \frac{\pi}{2}$, Equation (15) holds (cf. line 7 in **Algorithm 2**):

$$\alpha = (\boldsymbol{q}^*_t - \boldsymbol{q}^*_{t-1})^T (\boldsymbol{q}^+_t - \boldsymbol{q}^+_{t-1}) < 0 \tag{15}$$

where $\boldsymbol{q}^+_t$ is the currently observed configuration, $\boldsymbol{q}^+_{t-1}$ is the previously observed one, $\boldsymbol{q}^*_t$ is the generated target and $\boldsymbol{q}^*_{t-1}$ is the

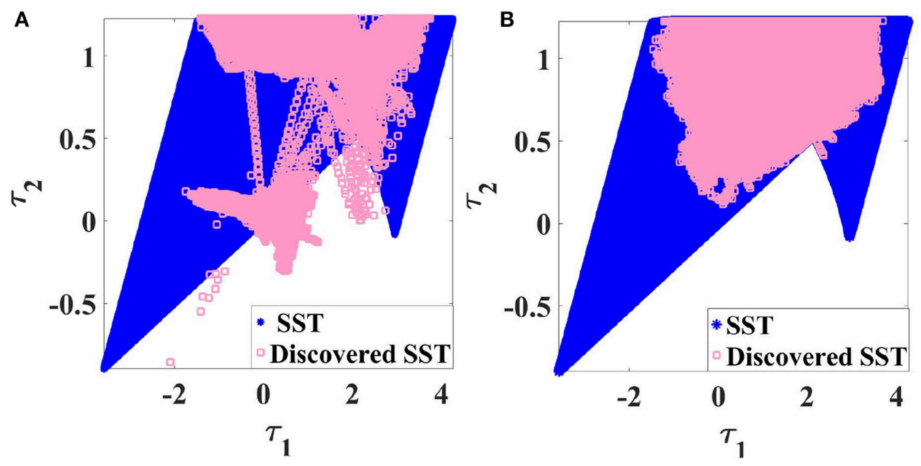


FIGURE 7 | Discovered **SST** of a 2R planar manipulator with specific joint limits **(A)** with original Goal Babbling, **(B)** with Constrained Goal Babbling.

Algorithm 2 Constrained Direction Sampling

```
 1: procedure DS( $\tau^{home}$, $q^{home}$, SST)
 2:     initialize the learner with $\hat{G}(q^{home}) = \tau^{home}$
 3:     choose random direction $\Delta q$
 4:     for $N$ number of samples
 5:         generate target $q_t^*$ according to Equation (13)
 6:         GBSCHEME($q_t^*$)
 7:         if $\alpha < 0$
 8:             go to $q_{t-1}^+$
 9:             choose new direction $\Delta q$
10:         end if
11:     end for
12: end procedure
13: procedure GBSCHEME($q_t^*$)
14:     estimate static torque $\hat{\tau}_t^*$ required to maintain $q_t^*$
15:     add exploratory noise $\sigma$:
            $\tau_t^+ = \hat{\tau}_t^* + \sigma(q_t^*, t)$
16:     if $\tau_t^+ \notin$ SST
17:         set $\tau_t^+ = \tau_m$ where $\{\tau_m \in \mathcal{T}_s : \forall \tau_n \in \mathcal{T}_s$
              $dist(\tau_t^+, \tau_m) \leqslant dist(\tau_t^+, \tau_n)\}$
18:     end if
19:     execute $\tau_t^+$ and observe $q_t^+$
20:     compute weight $w_t^{dir}$
21:     learner $\longleftarrow (\tau_t^+, q_t^+, w_t^{dir})$
22: end procedure
```

previously generated one. In this case, the agent will return to its previous configuration q_{t-1}^+ to avoid drifting and start following a new randomly selected direction again (Rolf, 2013; Rayyes and Steil, 2016).

One criterion of the weighting scheme, which has been previously proposed in Rolf et al. (2011), is adopted in order to favor training samples:

$$w_t^{dir} = \frac{1}{2}(1 + cos\sphericalangle(q_t^* - q_{t-1}^*, q_t^+ - q_{t-1}^+)) \tag{16}$$

w_t^{dir} is the direction criterion which evaluates whether the observed configuration and the generated target align well. This speeds up learning along the desired direction which is favorable in goal-directed algorithms. However, other weighting schemes could be selected as well.

5.3. Local Linear Map

As an incremental regression mechanism is required for online learning, a Local Linear Map (LLM) (Ritter, 1991) is employed. However, some modifications are necessary for exploiting symmetries. In this case, the learner must deal with scattered samples. Due to the initialization techniques of the standard LLM, receiving non-neighboring samples results in inconsistent outcomes. A further modification to gain more efficiency and reduce the number of required samples is proposed.

We will first explain the standard LLM algorithm for learning ISMs, and then the proposed modifications:

5.3.1. LLM for Learning ISMs

The inverse estimate $\hat{G}(q)$ is initialized with a first local linear function $\hat{G}^{(1)}(q)$ which is centered around a prototype vector $q_p{}^{(1)} = q^{home}$ corresponding to the initial static torque τ^{home}. M different new local linear functions $\hat{G}^{(i)}(q)$ are added incrementally during learning, centered around prototype vectors $q_p{}^{(i)}$ and active only if new data is received in their close vicinity determined by a radius d. Let ϱ_i denote a local configuration vector given by Equation (17):

$$\varrho_i = \left(\frac{q^* - q_p{}^{(i)}}{d}\right) \tag{17}$$

The inverse estimate $\hat{G}(q)$ is updated continuously and comprises a weighted linear sum of the linear functions $\hat{G}^{(i)}(\varrho_i)$. The weights are given by a Gaussian responsibility function $GR(q)$ as shown in Equation (18).

$$\left.\begin{aligned}\hat{\boldsymbol{G}}(\boldsymbol{q}^*) &= \frac{1}{N(\boldsymbol{q}^*)}\sum_{i=1}^{M} GR(\boldsymbol{\varrho}_i)\cdot \hat{\boldsymbol{G}}^{(i)}(\boldsymbol{\varrho}_i)\\ GR(\boldsymbol{\varrho}) &= \exp\left(-||\boldsymbol{\varrho}||^2\right)\\ N(\boldsymbol{q}^*) &= \sum_{i=1}^{M} GR(\boldsymbol{\varrho}_i)\\ \hat{\boldsymbol{G}}^{(i)}(\boldsymbol{\varrho}_i) &= \boldsymbol{W}^{(i)}\cdot \boldsymbol{\varrho}_i + \boldsymbol{o}^{(i)},\end{aligned}\right\} \tag{18}$$

$N(\boldsymbol{q}^*)$ normalizes the Gaussian responsibility functions in the inverse estimate.

The first linear function $\hat{\boldsymbol{G}}^{(1)}(\boldsymbol{q})$ is initialized with $\boldsymbol{q_p}^{(1)} = \boldsymbol{q}^{home}$, $\boldsymbol{o}^{(1)} = \boldsymbol{\tau}^{home}$, $\boldsymbol{W}^{(1)} = \boldsymbol{0}$, and $\hat{\boldsymbol{G}}^{(1)}(\boldsymbol{q}) = \boldsymbol{\tau}^{home}$. A new local linear function $\hat{\boldsymbol{G}}^{(i+1)}(\boldsymbol{q})$ will be added when the learner receives a new training sample $\boldsymbol{q}_{new}$ at distance of at least d to all existing prototypes (i.e., $dist(\boldsymbol{q}_{new}, \boldsymbol{q_p}^{(i)}) \geqslant d$). The corresponding prototype vector is added ($\boldsymbol{q_p}^{(i+1)} = \boldsymbol{q}_{new}$). The offset $\boldsymbol{o}^{(i+1)}$ of $\hat{\boldsymbol{G}}^{(i+1)}(\boldsymbol{q})$ is initialized with the inverse estimate before adding the new function in order to avoid abrupt changes in the inverse estimate function, i.e., the insertion of the new function will not change the local behavior of $\hat{\boldsymbol{G}}(\boldsymbol{q})$ at $\boldsymbol{q}_{new}$. The weighting matrix $\boldsymbol{W}^{(i+1)}$ represents the slope of the linear function after inserting the new sample:

$$\left.\begin{aligned}\boldsymbol{o}^{(i+1)} &= \hat{\boldsymbol{G}}(\boldsymbol{q}_{new}).\\ \boldsymbol{W}^{(i+1)} &= \frac{\partial \hat{\boldsymbol{G}}(\boldsymbol{q})}{\partial \boldsymbol{q}} = J(\boldsymbol{q})\end{aligned}\right\} \tag{19}$$

where $J(\boldsymbol{q})$ is the Jacobian matrix of the inverse estimate (Rolf et al., 2011).

The parameter update is done at each step using a gradient descent with learning rate η in order to minimize the weighted squared error E_t given in Equation (21) as following:

$$\left.\begin{aligned}\boldsymbol{W}_{t+1}^{(i)} &= \boldsymbol{W}_t^{(i)} - \eta\cdot\frac{\partial E_t}{\partial \boldsymbol{W}^{(i)}}\\ \boldsymbol{o}_{t+1}^{(i)} &= \boldsymbol{o}_t^{(i)} - \eta\cdot\frac{\partial E_t}{\partial \boldsymbol{o}^{(i)}}\end{aligned}\right\} \tag{20}$$

$$E_t = w_t^{dir}\|\boldsymbol{\tau}_t^+ - \hat{\boldsymbol{\tau}}_t^+\|^2 \tag{21}$$

Note that the execution of $\boldsymbol{\tau}_t^+$ will result in $\boldsymbol{q}_t^+$ and the corresponding torque estimated by the learner for $\boldsymbol{q}_t^+$ is denoted by $\hat{\boldsymbol{\tau}}_t^+$. Hence, the goal is to minimize the error between the executed and the estimated torques in order to improve the estimation accuracy.

The connections between the prototypes are organized and distributed based on an Instantaneous Topological Map (ITM) described in Jockusch and Ritter (1999) which is particularly suited to online map construction.

5.3.2. LLM Modifications

In this paper, two main modifications are implemented:

First, if the received new sample has a distance $>2d$ to all existing prototypes, That causes a disproportionate change in the inverse estimate results due to the initialization techniques when inserting new functions (cf. Equation 19). The standard LLM therefore failed to approximate the model because of receiving non-neighboring samples when utilizing symmetries. To avoid such situations, the added function will be initialized with the new sample as given in Equation (22):

$$\left.\begin{aligned}\boldsymbol{o}^{(i+1)} &= \boldsymbol{\tau}_{new}\\ \boldsymbol{W}^{(i+1)} &= \boldsymbol{0}\end{aligned}\right\} \tag{22}$$

Second, the LLM approach updates the inverse estimate instantaneously and it therefore requires a lot of samples to converge. However, data acquisition is very costly in terms of time, tear, and wear. In order to reduce the number of required samples, multiple gradient descent steps are performed for each new sample until the error E_t stabilizes. Hence, each training sample has more influence on the inverse estimate update, and consequently, the number of required samples is reduced significantly.

5.4. The General Scheme for Symmetry Discovery and Learning ISMs

Figure 8 illustrates the required steps for symmetry discovery by generating torque profiles and for symmetry exploitation with online learning ISMs. In the discovery phase, first a target torque $\boldsymbol{\tau}$ is selected. Subsequently, **Algorithm 1** is applied to discover symmetric configurations. Multiple linear regression is then performed using the output of **Algorithm 1** to update the functional relations between primary symmetries. The applied torque profiles and observed joint angles are exploited to update the estimates of the **SST** and optionally the **BCTS** (cf. section 4.3). When a sufficient number of primary symmetries $n_{sym} \geq n_{min}$ of symmetries has been discovered, the learning phase begins and the functional relations between the primary symmetries are exploited to generate n_{sym} training samples based on one applied training torque vector. n_{min} is set here to the number of geometrical symmetries. Constrained Direction Sampling (cf. **Algorithm 2**) or any other online (or batch) learning approach can be applied to obtain the ISM. The learning phase is terminated if a desired validation error e_{max} (i.e., the torque RMSE threshold) is reached. e_{max} is determined based on the torque limits and the required accuracy for accomplishing the task. e_{val} is the training torque RMSE which is evaluated at each iteration (i.e., predefined number of samples) on randomly chosen training samples from the current iteration.

6. BATCH LEARNING

Lattice sampling is implemented to sample the **BCTS** and collect training data. A feed-forward network with n neurons in the hidden layer is implemented to learn ISMs in a batch learning fashion.

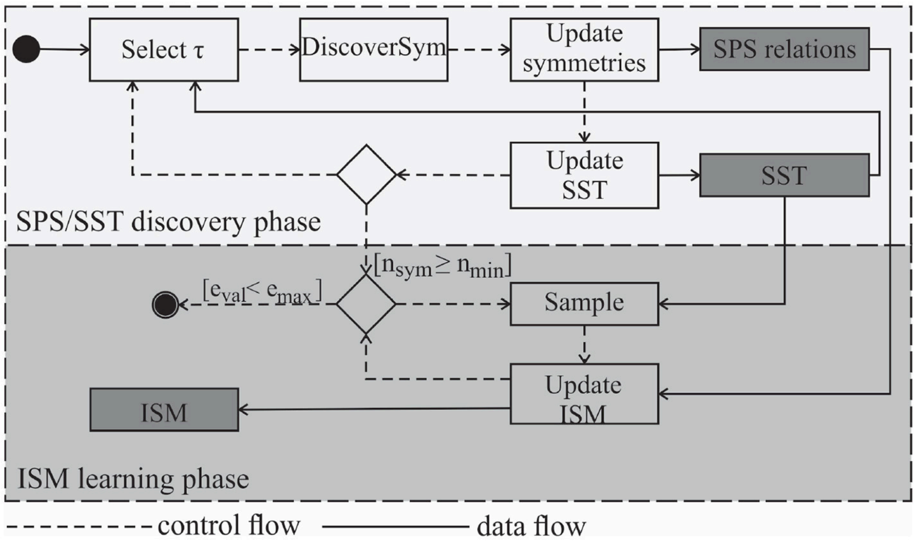


FIGURE 8 | Flowchart of the **SST** and SPS discovery as well as the ISM learning phase. The estimated **SST** is used to generate admissible torque samples and the SPS is used to generate n_{sym} training samples from one recorded sample.

A lattice $\mathcal{L}_s$ is the set of points which is characterized by an elementary unit cell. This elementary unit cell can be described by m vectors given in Equation (23) and is replicated over m-dimensional space.

$$\mathcal{L}_s = \sum_{i=1}^{m} \lambda_i \cdot \boldsymbol{p}_i, \quad 0 \leqslant \lambda_i \leqslant 1 \tag{23}$$

The vectors $\boldsymbol{p}_i$ are called also generators of the lattice (Cervellera et al., 2014).

7. EXPERIMENTAL RESULTS

This section presents experimental results for learning ISMs for a 2R planar robot and a 3R simplified human arm (Babiarz et al., 2015). The results show the efficiency gained by exploiting symmetries and demonstrate that exploiting symmetries is a generally applicable strategy which can be utilized with offline/online learning algorithms. Moreover, we demonstrate the efficiency gained by implementing LLM with multiple gradient descent steps (cf. section 5.3.2) for a 2R planar robot.

7.1. Exploiting Symmetries With Constrained Direction Sampling - Online Learning

7.1.1. 2R Planar Manipulator

Constrained Direction Sampling was employed to explore the **BCTS** and learn the ISM for the entire configuration space of the 2R planar robot (cf. **Figure 2A**) for which, each link length is 25 *cm*. **Figure 10** shows the learned area of the configuration space (blue area) by exploring merely the **BCTS** (red area) and exploiting the symmetries.

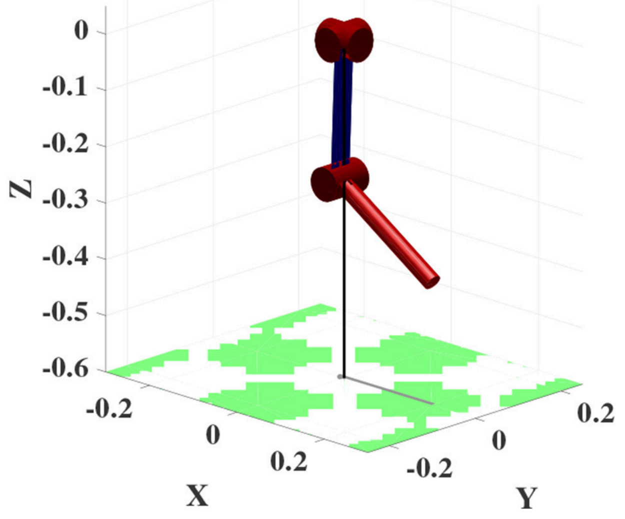


FIGURE 9 | Structure of the 3R simplified human arm with 25 *cm* link length.

After the training phase, the robot tries to reach and maintain 66 configuration targets regularly distributed on a grid in the **BCTS**. All targets were maintained well with an RMSE of 0.0053 *Nm* which represents the difference between the learner output, i.e., the estimated torque and the actual required static torque. Compared to the minimum and maximum static torques (−18.4, 24.5) *Nm* and (−6, 6.2) *Nm* for the first and second joints, respectively, the observed RMSE is negligibly small. **Figure 11** illustrates the results in the configuration space. The red crosses indicate the targets, and the blue circles represent the

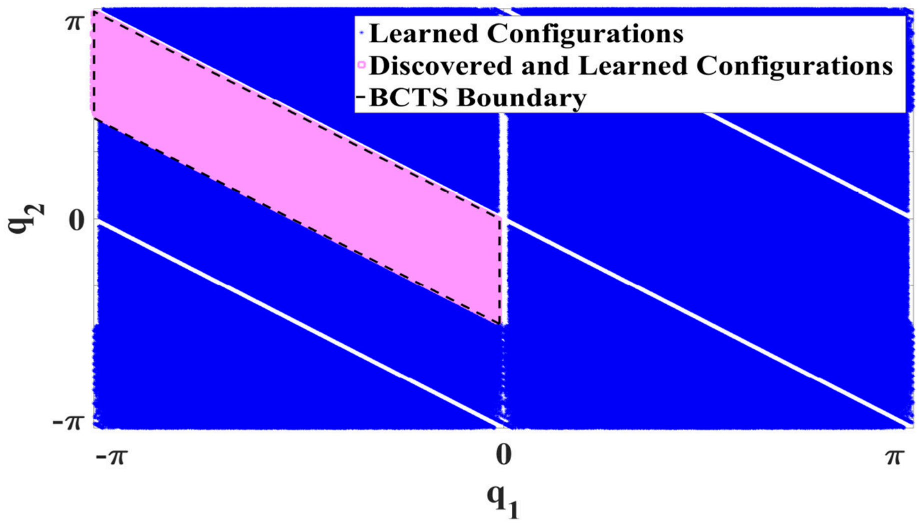


FIGURE 10 | Explored configurations (red) and learned configurations (blue) for the 2R robot by exploiting symmetries using Constrained Direction Sampling and LLM.

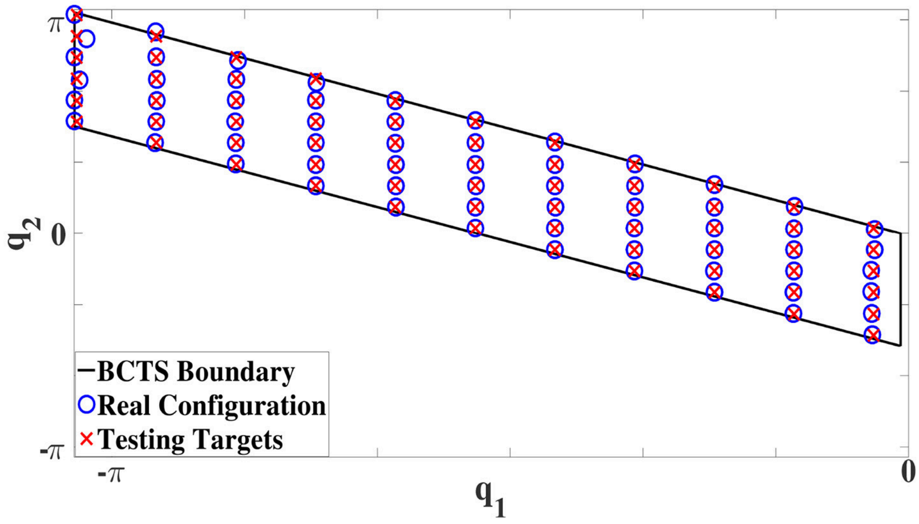


FIGURE 11 | Test performance for the 2R robot. The ISM is learned utilizing Constrained Direction Sampling and LLM with an RMSE of 0.0053 *Nm*. The boundary of the **BCTS** is indicated by the black parallelogram, the red crosses indicate the test targets, and the blue circles represent the observed configurations.

observed configurations which illustrate the good performance as well; the boundary of the **BCTS** is indicated by the black parallelogram. Subsequently, the robot tries to maintain another 90 targets scattered over the entire configuration space. The performance was also very good, the robot managed to achieve all targets very accurately with an RMSE of 0.0052 *Nm* as shown in **Figure 12**.

Efficiency gained by iterating gradient descent step in LLM: In the experiment, LLM with a single gradient descent step per sample was implemented first with Constrained Direction Sampling. At least 540 iterations (each iteration consists of 100 samples) were required to discover the entire **BCTS** and achieve an RMSE of 0.0053 *Nm*. By increasing the number of iterations, the performance accuracy is increased as shown in **Figure 13**. The blue line represents the RSME of the torque evaluated for different numbers of iterations. The RMSE was 0.0024 *Nm* after 3000 iterations.

A significant reduction in the number of required samples was observed by iterating multiple gradient descent steps in LLM (LLM$_{it}$) with Constrained Direction Sampling. Only 30 iterations were required to learn the ISM and achieve the same accuracy, i.e., test RMSE of 0.0053 *Nm*. Hence, the number of required samples are decreased by a factor of 18. The robot performance is tested on 84 targets scattered over the entire configuration space as shown in **Figure 12B**.

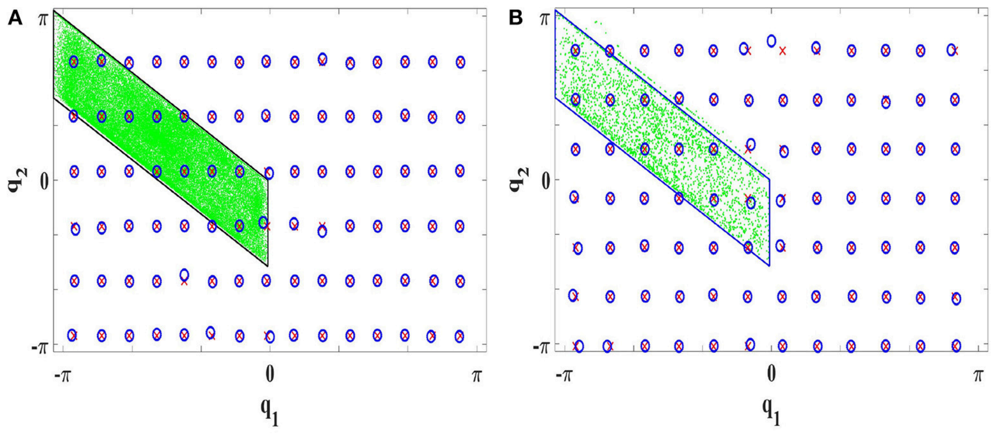


FIGURE 12 | Constrained Direction Sampling results for the 2R planar robot utilizing **(A)** LLM with 540 iterations **(B)** LLM$_{it}$ with 30 iterations. Torque RMSE is 0.0052 *Nm*. The green area is the discovered **BCTS**, the red crosses are the test targets, and the blue circles represent the real observed configurations.

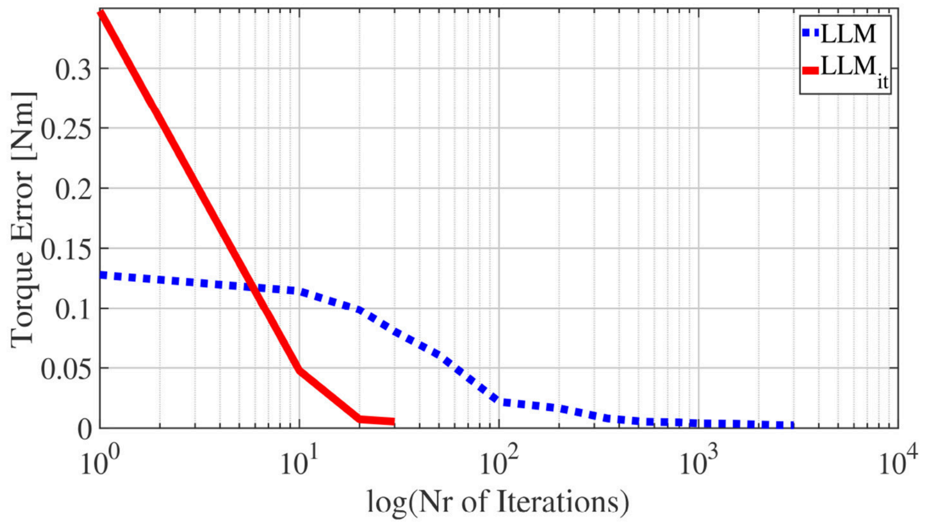


FIGURE 13 | Torque RMSE for Constrained Direction Sampling with LLM$_{it}$ (red) and LLM (blue).

The average training time required in each iteration for updating the LLM$_{it}$ is 3 *min* and 0.2 *min* for the LLM. Hence, the time cost per iteration for LLM$_{it}$ is 15 times higher. However, LLM requires 18 times the number of samples required for LLM$_{it}$. As data acquisition is costly and moving the robot to the sampled configurations is very time-consuming, the overall efficiency with LLM$_{it}$ is much higher than with LLM.

The torque RMSEs for different numbers of iterations (red line) are shown in **Figure 13**. As we can see from the figure, the torque RMSE converges much faster for LLM$_{it}$ than LLM.

7.1.2. 3R Robot Arm

Constrained Direction Sampling with LLM$_{it}$ is implemented to learn the ISM for the 3R manipulator (cf. **Figure 9**). After exploring the **BCTS**, the robot performance is tested on 64 targets regularly distributed on a grid in the configuration space. At least 140 iterations were required to achieve an RMSE of 0.26 *Nm*. The minimum and maximum torques for the first, the second, and the third joints are (−24.4, 24) *Nm*, (−24.2, 24.2) *Nm*, and (−12.4, 12.2) *Nm*, respectively. The achieved accuracy is very good compared to the torque limits.The results are illustrated in the configuration space as shown in **Figure 15A**.

7.2. Exploiting Symmetries With Lattice Sampling - Batch Learning

7.2.1. 2R Planar Manipulator

To demonstrate the general applicability of symmetry exploitation, we investigate batch learning to learn the ISM of the 2R robot (cf. **Figure 2A**) based on a lattice sampling approach. Lattice sampling was performed to collect training samples in the **BCTS**. A feed-forward neural network with one hidden layer consisting of 18 neurons was used in a batch

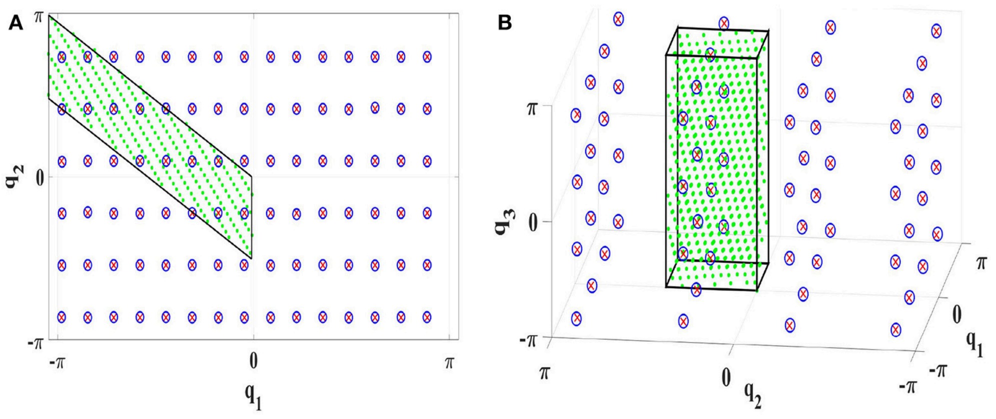


FIGURE 14 | Learning ISMs with exploiting symmetries with batch learning **(A)** for the 2R planar manipulator with an RMSE of 0.0051 *Nm* **(B)** for the 3R manipulator with an RMSE of 0.009 *Nm*. The green area represents the discovered **BCTS**, the border of the **BCTS** is indicated by the black lines, the test targets are visualized by red crosses and the blue circles indicate the real configurations.

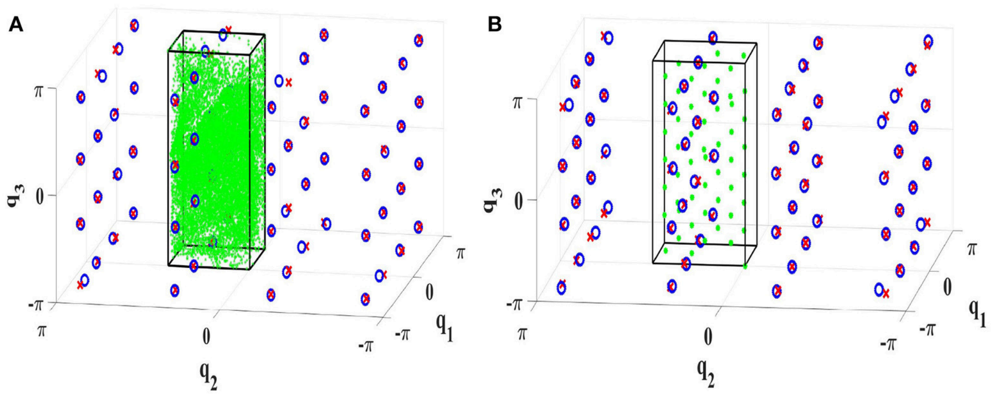


FIGURE 15 | Learning ISMs with exploiting symmetries **(A)** online with Constrained Direction Sampling **(B)** offline with Lattice Sampling. The torque RMSE is 0.028 *Nm*. The green area represents the discovered **BCTS**, the border of the **BCTS** is indicated by the black cube, the test targets are visualized by red crosses and the blue circles indicate the real configurations.

learning fashion. Only 255 samples in the **BCTS** were required to learn the ISM for the entire configuration space with almost the same testing torque RMSE of 0.0051 using the same 90 testing targets as in section 7.1.1. The result is illustrated in **Figure 14A**.

Lattice sampling was then performed for the entire configuration space without exploiting symmetries. 2040 samples were required to achieve approximately the same RMSE of 0.005 *Nm*. The number of required samples to learn the ISM of the 2R robot was reduced by a factor of 8 by exploiting primary symmetries. This factor corresponds well to the number of 8 primary symmetries for the 2R robot.

7.2.2. 3R Robot Manipulator

We did the same experiment as in section 7.1.2 utilizing lattice sampling and a feed-forward neural network with 18 neurons in the hidden layer in offline learning fashion. Only 65 training samples in the **BCTS** were required to achieve approximately the same accuracy with RMSE of 0.28 *Nm*. The good performance of the robot is also illustrated in **Figure 15B**. To illustrate the efficiency gained by using symmetries, Lattice sampling was implemented without exploiting symmetries. 855 samples were required to explore the entire configuration space with approximately the same RMSE of 0.03 *Nm*. The number of required samples to learn the ISM of the 3R robot was reduced by a factor of 16.13 which matches the number of 16 primary symmetries well. To achieve higher accuracy, 600 samples with 30 hidden neurons were required to achieve an RMSE of 0.009 *Nm*. The result is demonstrated in **Figure 14B**.

7.3. Discussion

The number of required samples to learn ISMs for 2R and 3R manipulators were reduced by a factor of 8 and 16, respectively, resulting from exploiting primary symmetries and constraining the exploration to the **BCTS** only. Hence, exploiting symmetries can drastically increase learning efficiency – regardless whether offline or online learning schemes are considered – by reducing the number of required samples by a factor which approximately equals the number of discovered primary symmetries in the presented experiments. Further efficiency gains can be expected if secondary symmetries are exploited as well.

Note that the number of samples in batch learning is lower than that required in the presented online learning approach. Nevertheless, even batch learning approaches can greatly benefit from a significant reduction in the number of required samples by exploiting symmetries. However, online learning techniques such as Goal Babbling and Direction Sampling, which generate targets on the fly and update the learner at each step simultaneously, best fit the concepts of gradual exploration as well as "learning while behaving" – hence they best reflect human developmental aspects in robot learning.

8. CONCLUSION AND OUTLOOK

We showed that inverse statics mappings of discretely-actuated serial manipulators can be learned very accurately, if the problems arising from exploratory learning in the torque domain are properly addressed. To learn ISMs online and from scratch, we constrained the Direction Sampling approach and improved the LLM learner. Naturally, these modifications may be useful also in other contexts and comprise a contribution to increase efficiency of any learning scheme employing these methods. Moreover, we demonstrated that the efficiency of learning inverse statics mappings can be further increased significantly by exploiting inherent symmetries of the mapping, a concept that we formalized properly and which as well is relevant beyond the particular exploratory learning application. To demonstrate its generality, we successfully integrated it into online Constrained Direction Sampling and a more standard batch learning approach based on lattice sampling. The presented results indicated that factors of at least 8 and 16 w.r.t. the number of samples can be achieved for a 2R and a 3R robot, respectively. Thus, exploiting symmetries is a promising strategy to increase the efficiency of learning both online and offline, and it is rather a general strategy and not limited to learning ISMs only, but it can be exploited in other functions or mappings.

We initially considered the particular problem of learning the inverse statics model as a rather simpler subproblem of the general inverse dynamics exploratory learning. However, it appears that it already displays some major difficulties of torque-based exploratory learning. And it requires substantial effort to be tackled. That led to the novel approaches on symmetries and the learning methods presented in this paper, which all have their right in itself and provide useful tools beyond the ISM learning alone. It is not obvious though, how to make the next step toward general inverse dynamics exploratory learning without relying on a pre-defined closed-loop controller, because that requires to suggest a general way to automatically choose target trajectories in the joint space that are safe, but representative and increasingly complex, while all other problems of efficiency and ambiguity still remain.

Currently, our approach is limited to primary symmetries as the functional relations between secondary symmetries prove to be challenging. Furthermore, elasticity as well as nonlinear friction effects are currently not considered. This sheds some light on more direct and natural extensions for future work, which we are working on. The proposed symmetry-based exploration is being (i) implemented in the real application, (ii) generalized to learn primary and secondary symmetries for discretely-actuated serial manipulators with arbitrary geometrical and inertial properties, (iii) extended to incorporate link and joint flexibility as well as nonlinear friction effects, which will pave the way for thorough experimental evaluation on a robot with variable stiffness actuators and (iv) implement a dictionary with a fixed budget to update LLM using a sub-data set instead of the current sample only. Furthermore, due to the same dimensionality of action and observation spaces, the efficiency advantage of Goal Babbling is less pronounced for learning ISMs than learning IK. However, this disadvantage is partially compensated by the efficiency gained by exploiting symmetry properties of ISMs and limiting the exploration to **BCTS** only. In our recent work (Rayyes et al., 2018), we additionally lay the foundation for increasing the scalability by learning IK and the inverse statics IS_x and ISMs simultaneously. IS_x maps from Cartesian space to the motor space. Hence, ISMs can be inferred by relating IK and IS_X.

AUTHOR CONTRIBUTIONS

RR: Conception and design of the research, acquisition, analysis and interpretation of data; DK: Conception and design of the research, analysis and interpretation of data; JS: Conception and design of the research, analysis and interpretation of data. All authors writing of the paper.

FUNDING

RR receives funding from DAAD - Research Grants-Doctoral Programme in Germany scholarship. This publication is supported by the German Research Foundation and the Open Access Publication Funds in the Technische Univerität Braunschweig.

ACKNOWLEDGMENTS

We thank C. Hartmann for helping in symmetry discovery implementation.

REFERENCES

Akiyama, T., Hachiya, H., and Sugiyama, M. (2010). Efficient exploration through active learning for value function approximation in reinforcement learning. *Neural Netw.* 23, 639–648. doi: 10.1016/j.neunet.2009.12.010

Asada, M.,MacDorman, K. F., Ishiguro, H., and Kuniyoshi, Y. (2001). Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Rob. Auton. Syst.* 37, 185–193. doi: 10.1016/S0921-8890(01)00157-9

Babiarz, A., Czornik, A., Klamka, J., and Niezabitowski, M. (2015). Dynamics modeling of 3d human arm using switched linear systems. *Asian Conference* *on Intelligent Information and Database Systems* (Cham: Springer), vol. 9012, 258–267.

Baranes, A., and Oudeyer, P. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robot. Auton. Syst.* 61, 49–73. doi: 10.1016/j.robot.2012.05.008

Cervellera, C., Gaggero, M., Macciò, D., and Marcialis, R. (2014). "Lattice sampling for efficient learning with nadaraya-watson local models," in *2014 International Joint Conference on Neural Networks (IJCNN)* (Beijing: IEEE), 1915–1922.

Craig, J. (1986). *Introduction to Robotics: Mechanics & Control*. Boston, MA: Addison-Wesley Publishing.

De Luca, A., and Panzieri, S. (1994). An iterative scheme for learning gravity compensation in flexible robot arms. *Automatica* 30, 993–1002.

De Luca, A., and Panzieri, S. (1996). End-effector regulation of robots with elastic elements by an iterative scheme. *Int. J. Adapt. Control Signal Proc.* 10, 379–393.

Demiris, Y., and Meltzoff, A. (2008). The robot in the crib: a developmental analysis of imitation skills in infants and robots. *Infant Child Dev.* 17, 43–53. doi: 10.1002/icd.543

Draper, N. R., and Smith, H. (1998). *Applied Regression Analysis*. New York, NY: Wiley.

D'Souza, Vijayakumar, S., and Schaal, S. (2001). Learning inverse kinematics. *Int. Conf. Intell. Rob. Syst.* 1, 298–303. doi: 10.1109/IROS.2001.973374

Forestier, S., and Oudeyer, P. (2016). "Modular active curiosity-driven discovery of tool use," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016* (Daejeon), 3965–3972.

Giorelli, M., Renda, F., Calisti, M., Arienti, A., Ferri, G., and Laschi, C. (2015). Neural network and jacobian method for solving the inverse statics of a cable-driven soft arm with nonconstant curvature. *IEEE Trans. Robot.* 31, 823–834. doi: 10.1109/TRO.2015.2428511

Gomi, H., and Kawato, M. (1993). Neural network control for a closed-loop system using feedback-error-learning. *Neural Netw.* 6, 933–946. doi: 10.1016/S0893-6080(09)80004-X

Jockusch, J., and Ritter, H. (1999). "An instantaneous topological mapping model for correlated stimuli," in *Neural Networks, 1999. IJCNN '99. International Joint Conference* (Washington, DC), vol. 1, 529–534.

Jordan, M., and Rumelhart, D. (1992). Forward models: supervised learning with a distal teacher. *Cogn. Sci.* 16, 307–354.

Loviken, P., and Hemion, N. (2017). Online-learning and planning in high dimensions with finite element goal babbling. *Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)* (Lisbon).

Luca, A. D., and Panzieri, S. (1993). Learning gravity compensation in robots: rigid arms, elastic joints, flexible links. *Int. J. Adapt. Control Signal Proc.* 7, 417–433.

Meier, F., Kappler, D., Ratliff, N., and Schaal, S. (2016). "Towards robust online inverse dynamics learning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Daejeon), 4034–4039.

Moulin-Frier, C., Nguyen, S. M., and Oudeyer, P.-Y. (2013). Self-organization of early vocal development in infants and machines: The role of intrinsic motivation. *Front. Psychol.* 4:1006. doi: 10.3389/fpsyg.2013.01006

Peters, J., and Schaal, S. (2008). Learning to control in operational space. *Int. J. Robot. Res.* 27, 197–212. doi: 10.1177/0278364907087548

Philippsen, A. K., Reinhart, R. F., and Wrede, B. (2016). "Goal babbling of acoustic-articulatory models with adaptive exploration noise," in *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)* (Cergy-Pontoise: IEEE), 72–78.

Rayyes, R., Kubus, D., and Steil, J. (2018). "Multi-stage goal babbling for learning inverse models simultaneously," in *IROS Workshop 2018, BODIS: The Utility of Body, Interaction and Self Learning in Robotics Workshop* (Madrid).

Rayyes, R., and Steil, J. J. (2016). "Goal babbling with direction sampling for simultaneous exploration and learning of inverse kinematics of a humanoid robot," in *Proceedings of the Workshop on New Challenges in Neural Computation*, Vol. 4 (Hanover), 56–63.

Ritter, H. (1991). "Learning with the self-organizing map," in *Artificial Neural Networks : Proceedings of the 1991 International Conference on Artificial Neural Networks [ICANN-91]*, vol. 1, ed T. Kohonen (Espoo), 379–384.

Rolf, M. (2013). "Goal babbling with unknown ranges: a direction-sampling approach," in *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)* (Osaka), 1–7.

Rolf, M., and Steil, J. (2014). Efficient exploratory learning of inverse kinematics on a bionic elephant trunk. *IEEE Trans. Neural Netw. Learn. Syst.* 25, 1147–1160. doi: 10.1109/TNNLS.2013.2287890

Rolf, M., Steil, J. J., and Gienger, M. (2010). Goal babbling permits direct learning of inverse kinematics. *IEEE Trans. Auton. Mental Dev.* 2, 216–229. doi: 10.1109/TAMD.2010.2062511

Rolf, M., Steil, J. J., and Gienger, M. (2011). "Online goal babbling for rapid bootstrapping of inverse models in high dimensions," in *IEEE International Conference on Development and Learning (ICDL)* (Frankfurt am Main), 1–8.

Şimşek, O., and Barto, A. G. (2006). "An intrinsic reward mechanism for efficient exploration," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (New York, NY: ACM), 833–840.

Thuruthel, T. G., Falotico, E., Cianchetti, M., and Laschi, C. (2016a). "Learning global inverse kinematics solution for a continuum robot" in *Robot Design and Control. ROMANSY21*, Vol. 569, eds V. Parenti-Castelli and W. Schiehlen (Cham: Springer), 47–54.

Thuruthel, T. G., Falotico, E., Cianchetti, M., Renda, F., and Laschi, C. (2016b). "Learning global inverse statics solution for a redundant soft robot," in *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics* (Lisbon: SciTePress), 303–310.

Vijayakumar, S., D'Souza, A., and Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Comput.* 17, 2602–2634. doi: 10.1162/089976605774320557

von Hofsten, C. (1982). *Eye Hand Coordination in the Newborn*. Washington, DC: American Psychological Association, 450–461.

Wolpert, D., and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Netw.* 11, 1317–1329.

Wolpert, D., Miall, R. C., and Kawato, M. (1998). Internal models in the cerebellum. *Trends Cognit. Sci.* 2, 338–347.

Xie, M., Zhong, Z. W., Zhang, L., Yang, H. J., Song, C. S., Li, J., et al. (2008). Self learning of gravity compensation by loch humanoid robot. *International Conference on Humanoid Robots* (Daejeon), 320–325.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH
published: 28 November 2018
doi: 10.3389/fnbot.2018.00078

Lifelong Learning of Spatiotemporal Representations With Dual-Memory Recurrent Self-Organization

German I. Parisi[1]*, *Jun Tani*[2], *Cornelius Weber*[1] *and Stefan Wermter*[1]

[1] *Knowledge Technology, Department of Informatics, Universität Hamburg, Hamburg, Germany,* [2] *Cognitive Neurorobotics Research Unit, Okinawa Institute of Science and Technology, Okinawa, Japan*

Artificial autonomous agents and robots interacting in complex environments are required to continually acquire and fine-tune knowledge over sustained periods of time. The ability to learn from continuous streams of information is referred to as *lifelong learning* and represents a long-standing challenge for neural network models due to *catastrophic forgetting* in which novel sensory experience interferes with existing representations and leads to abrupt decreases in the performance on previously acquired knowledge. Computational models of lifelong learning typically alleviate catastrophic forgetting in experimental scenarios with given datasets of static images and limited complexity, thereby differing significantly from the conditions artificial agents are exposed to. In more natural settings, sequential information may become progressively available over time and access to previous experience may be restricted. Therefore, specialized neural network mechanisms are required that adapt to novel sequential experience while preventing disruptive interference with existing representations. In this paper, we propose a dual-memory self-organizing architecture for lifelong learning scenarios. The architecture comprises two growing recurrent networks with the complementary tasks of learning object instances (episodic memory) and categories (semantic memory). Both growing networks can expand in response to novel sensory experience: the episodic memory learns fine-grained spatiotemporal representations of object instances in an unsupervised fashion while the semantic memory uses task-relevant signals to regulate structural plasticity levels and develop more compact representations from episodic experience. For the consolidation of knowledge in the absence of external sensory input, the episodic memory periodically replays trajectories of neural reactivations. We evaluate the proposed model on the CORe50 benchmark dataset for continuous object recognition, showing that we significantly outperform current methods of lifelong learning in three different incremental learning scenarios.

Keywords: lifelong learning, complementary learning systems, self-organizing networks, continuous object recognition, catastrophic forgetting

OPEN ACCESS

Edited by:
Gianluca Baldassarre,
Consiglio Nazionale Delle Ricerche (CNR), Italy

Reviewed by:
Guenther Palm,
Universität Ulm, Germany
Andreas Knoblauch,
Hochschule Albstadt-Sigmaringen, Germany
Jochen Triesch,
Goethe-Universität Frankfurt am Main, Germany

****Correspondence:***
German I. Parisi
parisi@informatik.uni-hamburg.de

Received: *29 May 2018*
Accepted: *06 November 2018*
Published: *28 November 2018*

Citation:
Parisi GI, Tani J, Weber C and Wermter S (2018) Lifelong Learning of Spatiotemporal Representations With Dual-Memory Recurrent Self-Organization. Front. Neurorobot. 12:78. doi: 10.3389/fnbot.2018.00078

1. INTRODUCTION

Artificial autonomous agents and robots interacting in dynamic environments are required to continually acquire and fine-tune their knowledge over time (Thrun and Mitchell, 1995; Parisi et al., 2018a). The ability to progressively learn over a sustained time span by accommodating novel knowledge while retaining previously learned experiences is referred to as *continual* or *lifelong learning*. In contrast to state-of-the-art deep learning models that typically rely on the full training set being available at once (see LeCun et al., 2015 for a review), lifelong learning systems must account for situations in which the training data become incrementally available over time. Effective models of lifelong learning are crucial in real-world conditions where an autonomous agent cannot be provided with all the necessary prior knowledge to interact with the environment and the direct access to previous experience is restricted (Thrun and Mitchell, 1995). Importantly, there may be no distinction between training and test phases, which requires the system to concurrently learn and timely trigger behavioral responses (Cangelosi and Schlesinger, 2015; Tani, 2016).

Lifelong machine learning represents a long-standing challenge due to *catastrophic forgetting* or *interference*, i.e., training a model with a new task leads to an abrupt decrease in the performance on previously learned tasks (McCloskey and Cohen, 1989). To overcome catastrophic forgetting, computational models must adapt their existing representations on the basis of novel sensory experience while preventing disruptive interference with previously learned representations. The extent to which a system must be flexible for learning novel knowledge and stable for preventing the disruption of consolidated knowledge is known as the *stability-plasticity dilemma*, which has been extensively studied for both computational and biological systems (e.g., Grossberg, 1980, 2007; Mermillod et al., 2013; Ditzler et al., 2015).

Neurophysiological evidence suggests distributed mechanisms of structural plasticity that promote lifelong memory formation, consolidation, and retrieval in multiple brain areas (Power and Schlaggar, 2016; Zenke et al., 2017a). Such mechanisms support the development of the human cognitive system on the basis of sensorimotor experiences over sustained time spans (Lewkowicz, 2014). Crucially, the brain must constantly perform two complementary tasks: (i) recollecting separate episodic events (specifics), and (ii) learning the statistical structure from the episodic events (generalities). The complementary learning systems (CLS) theory (McClelland et al., 1995; Kumaran et al., 2016) holds that these two interdependent operations are mediated by the interplay of the mammalian hippocampus and neocortex, providing the means for *episodic memory* (specific experience) and *semantic memory* (general structured knowledge). Accordingly, the hippocampal system exhibits quick learning of sparse representations from episodic experience which will, in turn, be transferred and integrated into the neocortical system characterized by a slower learning rate with more compact representations of statistical regularities.

Re-training a (deep) neural architecture from scratch in response to novel sensory input can require extensive computational effort. Furthermore, storing all the previously encountered data in lifelong learning scenarios has the general drawback of large memory requirements. Instead, Robins (1995) proposed *pseudo-rehearsal* (or *intrinsic replay*) in which previous memories are revisited without the need of explicitly storing data samples. *Pseudo-samples* are drawn from a probabilistic or generative model and replayed to the system for memory consolidation. From a biological perspective, the direct access to past experiences is limited or restricted. Therefore, the replay of hippocampal representations in the absence of external sensory input plays a crucial role in memory encoding (Carr et al., 2011; Kumaran et al., 2016). Memory replay is argued to occur through the reactivation of neural patterns during both sleep and awake states (e.g., free recall; Gelbard-Sagiv et al., 2008). Hippocampal replay provides the means for the gradual integration of knowledge into neocortical structures through the reactivation of recently acquired knowledge interleaved with the exposure to ongoing episodic experience (McClelland et al., 1995). Consequently, the periodic replay of previously encountered samples can alleviate catastrophic forgetting during incremental learning tasks, especially when the number of training samples for the different classes is unbalanced or when a sample is encountered only once (Robins, 1995).

A number of computational approaches have drawn inspiration from the learning principles observed in biological systems. Machine learning models addressing lifelong learning can be divided into approaches that regulate intrinsic levels of plasticity to protect consolidated knowledge, that dynamically allocate neural resources in response to novel experience, or that use complementary dual-memory systems with memory replay (see section 2). However, most of these methods are designed to address supervised learning on image datasets of very limited complexity such as MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky, 2009) while not scaling up to incremental learning tasks with larger-scale datasets of natural images and videos (Kemker et al., 2018; Parisi et al., 2018a). Crucially, such models do not take into account the temporal structure of the input which plays an important role in more realistic learning conditions, e.g., an autonomous agent learning from the interaction with the environment. Therefore, in contrast to approaches in which static images are learned and recognized in isolation, we focus on lifelong learning tasks where sequential data with meaningful temporal relations become progressively available over time.

In this paper, we propose a growing dual-memory (GDM) architecture for the lifelong learning of spatiotemporal representations from videos, performing continuous object recognition at an instance level (episodic knowledge) and at a category level (semantic knowledge). The architecture comprises two recurrent self-organizing memories that dynamically adapt the number of neurons and synapses: the episodic memory learns representations of sensory experience in an unsupervised fashion through input-driven plasticity, whereas the semantic memory develops more compact representations of statistical regularities embedded in episodic experience. For this purpose, the semantic memory receives neural activation trajectories from the episodic memory and uses task-relevant signals (annotated labels) to modulate levels of neurogenesis and neural update. Internally generated neural activity patterns in the episodic memory are periodically replayed to both memories in the absence of sensory input, thereby mitigating catastrophic forgetting during incremental learning. We conduct a series of experiments with the recently published Continuous Object Recognition (CORe50) benchmark dataset (Lomonaco and Maltoni, 2017). The dataset comprises 50 objects within 10 categories with image sequences captured under different conditions and containing multiple views of the same objects (indoors and outdoors, varying background, object pose, and degree of occlusion). We show that our model scales up to learning novel object instances and categories and that it outperforms current lifelong learning approaches in three different incremental learning scenarios.

2. RELATED WORK

The CLS theory (McClelland et al., 1995) provides the basis for computational frameworks that aim to generalize across experiences while retaining specific memories in a lifelong fashion. Early computational attempts include French (1997) who developed a dual-memory framework using pseudo-rehearsal (Robins, 1995) to transfer memories, i.e., the training samples are not explicitly kept in memory but drawn from a probabilistic model. However, there is no empirical evidence showing that this or similar contemporaneous approaches (see O'Reilly and Norman, 2002 for a review) scale up to large-scale image and video benchmark datasets. More recently, Gepperth and Karaoguz (2015) proposed two approaches for incremental learning using a modified self-organizing map (SOM) and a SOM extended with a short-term memory (STM). We refer to these two approaches as GeppNet and GeppNet+STM, respectively. In GeppNet, task-relevant feedback from a regression layer is used to select whether learning in the self-organizing hidden layer takes place. In GeppNet+STM, the STM is used to store novel knowledge which is occasionally played back to the GeppNet layer during sleep phases interleaved with training phases. This latter approach yields better performance and faster convergence in incremental learning tasks with the MNIST dataset. However, the STM has a limited capacity, thus learning new knowledge can overwrite old knowledge. In both cases, the learning process is divided into the initialization and the actual incremental learning phase. Furthermore, GeppNet and GeppNet+STM require storing the entire training dataset during training. Kemker and Kanan (2018) proposed the FearNet model for incremental class learning inspired by studies of memory recall and consolidation in the mammalian brain during fear conditioning (Kitamura et al., 2017). FearNet uses a hippocampal network capable of immediately recalling new examples, a PFC network for long-term memories, and a third neural network inspired by the basolateral amygdala for determining whether the system should use the PFC or hippocampal network for a particular example. FearNet consolidates information from its hippocampal network to its PFC network during sleep phases. Kamra et al. (2018) presented a similar dual-memory framework for lifelong learning that uses a variational autoencoder as a generative model for pseudo-rehearsal. Their framework generates a short-term memory module for each new task. However, prior to consolidation, predictions are made using an oracle, i.e., they know which module contains the associated memory.

Different methods have been proposed that are based on regularization techniques to impose constraints on the update of the neural weights. This is inspired by neuroscience findings suggesting that consolidated knowledge can be protected from interference via changing levels of synaptic plasticity (Benna and Fusi, 2016) and is typically modeled in terms of adding regularization terms that penalize changes in the mapping function of a neural network. For instance, Li and Hoiem (2016) proposed a convolutional neural network (CNN) architecture in which the network that predicts the previously learned tasks is enforced to be similar to the network that also predicts the current task by using knowledge distillation, i.e., the transferring of knowledge from a large, highly regularized model to a smaller model. This approach, known as learning without forgetting (LwF), has the drawbacks of highly depending on the relevance of the tasks and that the training time for one task linearly increases with the number of old tasks. Kirkpatrick et al. (2017) proposed elastic weight consolidation (EWC) which adds a penalty term to the loss function and constrains the weight parameters that are relevant to retain previously learned tasks. However, this approach requires a diagonal weighting over the parameters of the learned tasks which is proportional to the diagonal of the Fisher information metric, with synaptic importance being computed offline and limiting its computational application to low-dimensional output spaces. Zenke et al. (2017b) proposed to alleviate catastrophic forgetting by allowing individual synapses to estimate their importance for solving a learned task. Similar to Kirkpatrick et al. (2017), this approach penalizes changes to the most relevant synapses so that new tasks can be learned with minimal interference. In this case, the synaptic importance is computed in an online fashion over the learning trajectory in the parameter space.

In general, regularization approaches comprise additional loss terms for protecting consolidated knowledge which, with a limited amount of neural resources, leads to a trade-off on the performance of old and novel tasks. Other approaches expand the neural architecture to accommodate novel knowledge. Rusu et al. (2016) proposed to block any changes to the network trained on previous knowledge and expand the architecture by allocating novel sub-networks with a fixed capacity to be trained with the new information. This prevents catastrophic forgetting but leads the complexity of the architecture to grow with the number of learned tasks. Draelos et al. (2017) trained an autoencoder incrementally using the reconstruction error to show whether the older digits were retained. Their model added new neural units to the autoencoder to facilitate the addition of new MNIST digits. Rebuffi et al. (2017) proposed the iCaRL approach which stores example data points that are used along with new data to dynamically adapt the weights of a feature extractor. By combining new and old data, they prevent catastrophic forgetting but at the expense of a higher memory footprint.

The approaches described above are designed for the classification of static images, often exposing the learning algorithm to training samples in a random order. Conversely, in more natural settings, we make use of the spatiotemporal structure of the input. In previous research (Parisi et al., 2017), we showed that the lifelong learning of action sequences can be achieved in terms of prediction-driven neural dynamics with internal representations emerging in a hierarchy of recurrent self-organizing networks. The networks can dynamically allocate neural resources and update connectivity patterns according to competitive Hebbian learning by computing the input based on its similarity with existing knowledge and minimizing interference by creating new neurons whenever they are required. This approach has shown competitive results with batch learning methods on action benchmark datasets. However, the neural growth and update are driven by the minimization of the bottom-up reconstruction error and, thus, without taking into account top-down, task-relevant signals that can regulate the plasticity-stability balance. Furthermore, the model cannot learn in the absence of external sensory input, which leads to a non-negligible degree of disruptive interference during incremental learning tasks.

3. PROPOSED METHOD

The proposed architecture with growing dual-memory learning (GDM) comprises a deep convolutional feature extractor and two hierarchically arranged recurrent self-organizing networks (**Figure 1**). Both recurrent networks are extended versions of the Gamma-GWR model (Parisi et al., 2017) that dynamically create new neurons and connections in response to novel sequential input. The growing episodic memory (G-EM) learns from sensory experience in an unsupervised fashion, i.e., levels of structural plasticity are regulated by the ability of the network to predict the spatiotemporal patterns given as input. Instead, the growing semantic memory (G-SM) receives neural activation trajectories from G-EM and uses task-relevant signals (input annotations) to modulate levels of neurogenesis and neural update, thereby developing more compact representations of statistical regularities embedded in episodic experience. Therefore, G-EM and G-SM mitigate catastrophic forgetting through self-organizing learning dynamics with structural plasticity, increasing information storage capacity in response to novel input.

The architecture classifies image sequences at an instance level (episodic experience) and a category level (semantic knowledge). Thus, each input sample carries two labels which are used for the classification task at the different levels of the network hierarchy. For the consolidation of knowledge over time in the absence of sensory input, internally generated neural activity patterns in G-EM are periodically replayed to both memories, thereby mitigating catastrophic forgetting during incremental learning tasks. For this purpose, G-EM is equipped with synapses that learn statistically significant neural activity in the temporal domain. As a result, sequence-selective neural activation trajectories can be generated and replayed after each learning episode without explicitly storing sequential input.

3.1. Gamma-GWR

The Gamma-GWR model (Parisi et al., 2017) is a recurrent extension of the Grow-When-Required (GWR) self-organizing network (Marsland et al., 2002) that embeds a Gamma memory (Principe et al., 1994) for representing short-term temporal relations. The Gamma-GWR can dynamically grow or shrink in response to the sensory input distribution. New neurons will be created to better represent the input and connections (synapses) between neurons will develop according to competitive Hebbian learning, i.e., neurons that co-activate will be connected to each other. The Gamma-GWR learns the spatiotemporal structure of the input through the integration of temporal context into the computation of the self-organizing network dynamics.

The network is composed of a dynamic set of neurons, A, with each neuron consisting of a weight vector $\mathbf{w}_j$ and a number K of context descriptors $\mathbf{c}_{j,k}$ ($\mathbf{w}_j, \mathbf{c}_{j,k} \in \mathbb{R}^n$). Given the input $\mathbf{x}(t) \in \mathbb{R}^n$, the index of the best-matching unit (BMU), b, is computed as:

$$b = \arg\min_{j \in A}(d_j), \tag{1}$$

$$d_j = \alpha_0 \|\mathbf{x}(t) - \mathbf{w}_j\|^2 + \sum_{k=1}^{K} \alpha_k \|\mathbf{C}_k(t) - \mathbf{c}_{j,k}\|^2, \tag{2}$$

$$\mathbf{C}_k(t) = \beta \cdot \mathbf{w}_b^{t-1} + (1-\beta) \cdot \mathbf{c}_{b,k-1}^{t-1}, \tag{3}$$

where $\|\cdot\|^2$ denotes the Euclidean distance, α_i and β are constant factors that regulate the influence of the temporal context, $\mathbf{w}_b^{t-1}$ is the weight vector of the BMU at $t-1$, and $\mathbf{C}_k \in \mathbb{R}^n$ is the global context of the network with $\mathbf{C}_k(t_0) = 0$.

The activity of the network, $a(t)$, is defined in relation to the distance between the input and its BMU (Equation 2) as follows:

$$a(t) = \exp(-d_b), \tag{4}$$

thus yielding the highest activation value of 1 when the network can perfectly predict the input sequence (i.e., $d_b = 0$). Furthermore, each neuron is equipped with a habituation counter $h_j \in [0, 1]$ expressing how frequently it has fired based on a simplified model of how the efficacy of a habituating synapse reduces over time (Stanley, 1976). Newly created neurons start with $h_j = 1$. Then, the habituation counter of the BMU, b, and its neighboring neurons, n, iteratively decrease toward 0. The habituation rule (Marsland et al., 2002) for a neuron i is given by:

$$\Delta h_i = \tau_i \cdot \kappa \cdot (1 - h_i) - \tau_i, \tag{5}$$

with $i \in \{b, n\}$ and where τ_i and κ are constants that control the monotonically decreasing behavior. Typically, h_b is decreased faster than h_n with $\tau_b > \tau_n$.

The network is initialized with two neurons and, at each learning iteration, a new neuron is created whenever the activity

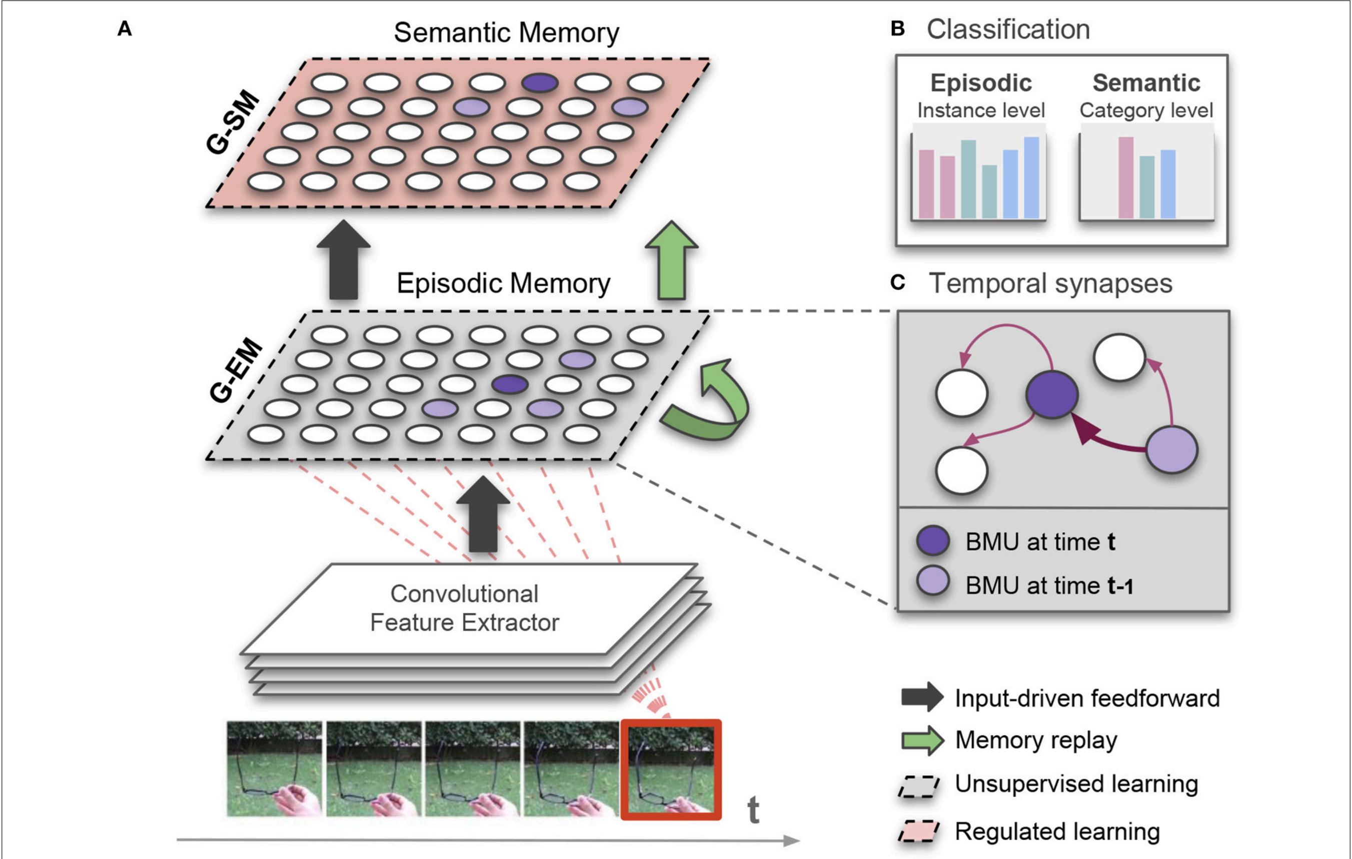


FIGURE 1 | (A) Illustration of our growing dual-memory (GDM) architecture for lifelong learning. Extracted features from image sequences are fed into a growing episodic memory (G-EM) consisting of an extended version of the recurrent Grow-When-Required network (section 3.2). Neural activation trajectories from G-EM are feed-forwarded to the growing semantic memory (G-SM) that develops more compact representations of episodic experience (section 3.3). While the learning process of G-EM remains unsupervised, G-SM uses class labels as task-relevant signals to regulate levels of structural plasticity. After each learning episode, internally generated neural activation trajectories are replayed to both memories (green arrows; section 3.4); **(B)** The architecture classifies image sequences at instance level (episodic experience) and at category level (semantic knowledge). For the purpose of classification, neurons in G-EM and G-SM associatively learn histograms of class labels from the input (red dashed lines); **(C)** To enable memory replay in the absence of sensory input, G-EM is equipped with temporal synapses that are strengthened (thicker arrow) between consecutively activated best-matching units (BMU).

of the network, $a(t)$, in response to the input $\mathbf{x}(t)$ is smaller than a given insertion threshold a_T. Furthermore, h_b must be smaller than a habituation threshold h_T in order for the insertion condition to hold, thereby fostering the training of existing neurons before new ones are added. The new neuron is created halfway between the BMU and the input. The training of the neurons is carried out by adapting the BMU b and the neurons n to which the b is connected:

$$\Delta \mathbf{w}_i = \epsilon_i \cdot h_i \cdot (\mathbf{x}(t) - \mathbf{w}_i), \tag{6}$$

$$\Delta \mathbf{c}_{i,k} = \epsilon_i \cdot h_i \cdot (\mathbf{C}_k(t) - \mathbf{c}_{i,k}), \tag{7}$$

with $i \in \{b, n\}$ and where ϵ_i is a constant learning rate ($\epsilon_n < \epsilon_b$). Furthermore, the habituation counters of the BMU and the neighboring neurons are updated according to Equation (5). Connections between neurons are updated on the basis of neural co-activation, i.e., when two neurons fire together (BMU and second-BMU), a connection between them is created if it does not exist. Each connection has an age that increases at each learning iteration. The age of the connection between the BMU and the second-BMU is reset to 0, whereas the other ages are increased by a value of 1. The connections with an age greater than a given threshold can be removed, and neurons without connections can be deleted.

For the purpose of classification, an associative matrix $H(j, l)$ stores the frequency-based distribution of sample labels during the learning phase, so that each neuron j stores the number of times that an input with label l had j as its BMU. As a result, the predicted label ξ_j for a neuron j can be computed as:

$$\xi_j = \arg\max_{l \in L} H(j, l), \tag{8}$$

where l is an arbitrary label. Therefore, the unsupervised Gamma-GWR can be used for classification without requiring the number of label classes to be predefined.

3.2. Episodic Memory

The learning process of growing episodic memory G-EM is unsupervised, thereby creating new neurons or updating existing ones to minimize the discrepancy between the sequential input and its neural representation. In this way, episodic memories can be acquired and fine-tuned iteratively through sensory experience. This is functionally consistent with hippocampal representations, e.g., in the dentate gyrus, which are responsible for pattern separation through the orthogonalization of incoming inputs supporting the auto-associative storage and retrieval of item-specific information from individual episodes (Yassa and Stark, 2011; Neunuebel and Knierim, 2014).

Given an input image frame, the extracted image feature vector (see section 4.1) is given as input to G-EM which recursively integrates the temporal context into the self-organizing neural dynamics. The spatial resolution of G-EM neurons can be tuned through the insertion threshold, a_T, with a greater a_T leading to more fine-grained representations since new neurons will be created whenever $a(t) < a_T$ (see Equation 4). The temporal depth is set by the number of context descriptors, K, with a greater K yielding neurons that activate for larger temporal windows (longer sequences), whereas the temporal resolution is set by the hyperparameters α and β (see Equations 2 and 3).

To enable memory replay in the absence of external sensory input, we extend the Gamma-GWR model by implementing temporal connections that learn trajectories of neural activity in the temporal domain. Such temporal connections are sequence-selective synaptic links which are incremented between two consecutively activated neurons (Parisi et al., 2016). Sequence selectivity driven by asymmetric connections has been argued to be a feature of the cortex (Mineiro and Zipser, 1998), where an active neuron pre-activates neurons encoding future patterns. Formally, when the two neurons i and j are consecutively activated at time $t-1$ and t, respectively, their temporal synaptic link $P_{(i,j)}$ is increased by $\Delta P_{(i,j)} = 1$. For each neuron $i \in A$, we can retrieve the next neuron v of a prototype trajectory by selecting

$$v = \arg\max_{j \in A \setminus i} P_{(i,j)}. \tag{9}$$

Recursively generated neural activation trajectories can be used for memory replay (see section 3.4). During the learning phase, G-EM neurons will store instance-level label classes ξ^I for the classification of the input (see Equation 8). Furthermore, since trajectories of G-EM neurons are replayed to G-SM in the absence of sensory input, G-EM neurons will also store labels at a category label l^C. Therefore, the associative matrix for each neuron j is of the form $H(j, l^I, l^C)$.

3.3. Semantic Memory

The growing semantic memory G-SM combines bottom-up drive from neural activity in G-EM and top-down signals (i.e., category-level labels from the input) to regulate structural plasticity levels. More specifically, the mechanisms of neurogenesis and neural weight update are regulated by the ability of G-SM to correctly classify its input. Therefore, while G-EM iteratively minimizes the discrepancy between the input sequences and their internal representations, G-SM will create new neurons only if the correct label of a training sample cannot be predicted by its BMU in G-SM. This is implemented as an additional constraint in the condition for neurogenesis so that new neurons are not created unless the predicted label of the BMU (Equation 8) does not match the input label.

G-SM receives as input activated neural weights from G-EM, i.e., the weight vector of a BMU in G-EM, $\mathbf{w}_b^{\mathrm{EM}}$, for a given input frame. As an additional mechanism to prevent novel sensory experience from interfering with consolidated representations, G-SM neurons are updated (Equations 6 and 7) only if the predicted label for the BMU in G-SM matches in the input label, i.e., if the BMU codes for the same object category as the input. In this way, the representations of an object category cannot be updated in the direction of the input belonging to a different category, which would cause disruptive interference.

As a result of hierarchical processing, G-SM neurons code for information acquired over larger temporal windows than neurons in G-EM. That is, one G-SM will fire for a number $K^{\mathrm{SM}} + 1$ of neurons fired in G-EM (where K^{SM} is the temporal depth of G-SM neurons). Since G-EM neurons will fire for a number $K^{\mathrm{EM}} + 1$ of input frames, G-SM neurons will code for a total of $K^{\mathrm{SM}} + K^{\mathrm{EM}} + 1$ input frames. This is consistent with established models of memory consolidation where neocortical representations code for information acquired over more extended time periods than the hippocampus (e.g., Kumaran and McClelland, 2012; Kumaran et al., 2016), thereby yielding a higher degree of temporal slowness.

Temporal slowness results from the statistical learning of spatiotemporal regularities, with neurons coding for prototype sequences of sensory experience. By using category-level signals to regulate neural growth and update, G-SM will develop more compact representations from episodic experience with neurons activating in correspondence of semantically-related input, e.g., the same neuron may activate for different instances of the same category and, because of the processing of temporal context, the same object seen from different angles. However, specialized mechanisms of slow feature analysis can be implemented that would yield invariance to complex input transformations such as view invariance (e.g., Berkes and Wiskott, 2005; Einhäuser et al., 2005). View invariance of objects is a prominent property of higher-level visual areas of the mammalian brain, with neurons coding for abstract representations of familiar objects rather than for individual views and visual features (Booth and Rolls, 1998; Karimi-Rouzbahani et al., 2017). Neurophysiological studies evidence that distributed representations in high-level visual regions of the neocortex (semantic) are less sparse than those of the hippocampus (episodic) and where related categories are represented by overlapping neural codes (Clarke and Tyler, 2014; Yamins et al., 2018).

3.4. Memory Replay

Hippocampal replay provides the means for the gradual integration of knowledge into neocortical structures and is thought to occur through the reactivation of recently acquired knowledge interleaved with the exposure to ongoing experiences (McClelland et al., 1995). Although the periodic replay of previous data samples can alleviate catastrophic forgetting, storing all previously encountered data samples has the general drawback of large memory requirements and large retraining computational times.

In pseudo-rehearsal (or intrinsic replay), memories are drawn from a probabilistic or generative model and replayed to the system for memory consolidation (Robins, 1995). In our case, however, we cannot simply draw or generate isolated and randomly selected pseudo-samples from a given distribution since we must account for preserving the temporal structure of the input. Therefore, we generate pseudo-patterns in terms of temporally-ordered trajectories of neural activity. For this purpose, we propose to use the asymmetric temporal links of G-EM (section 3.2) to recursively reactivate sequence-selective neural activity trajectories (RNATs) embedded in the network. RNATs can be computed for each neuron in G-EM for a given temporal window and replayed to G-EM and G-SM after each learning episode triggered by external input stimulation.

For each neuron j in G-EM, we generate a RNAT, S_j, of length $\lambda = K^{\mathrm{EM}} + K^{\mathrm{SM}} + 1$ as follows:

$$S_j = \langle \mathbf{w}^{\mathrm{EM}}_{\mathrm{s}(0)}, \mathbf{w}^{\mathrm{EM}}_{\mathrm{s}(1)}, ..., \mathbf{w}^{\mathrm{EM}}_{\mathrm{s}(\lambda)} \rangle, \tag{10}$$

$$\mathrm{s}(i) = \arg\max_{n \in A \setminus j} P_{(n,\mathrm{s}(i-1))}, i \in [1, \lambda], \tag{11}$$

where $P_{(i,j)}$ is the matrix of temporal synapses (as defined by Equation 9) and $\mathrm{s}(0) = j$. The class labels of the pseudo-patters in S_j can be retrieved according to Equation (8).

The set of generated RNATs from all G-EM neurons is replayed to G-EM and G-SM after each learning episode, i.e., a learning epoch over a batch of sensory observations. As a result of computing RNATs, sequence-selective prototype sequences can be generated and periodically replayed without the need of explicitly storing the temporal relations and labels of previously seen training samples. This is conceptually consistent with neurophysiological studies evidencing that hippocampal replay consists of the reactivation of previously stored patterns of neural activity occurring predominantly after an experience (Kudrimoti et al., 1999; Karlsson and Frank, 2009).

4. EXPERIMENTAL RESULTS

We perform a series of experiments evaluating the performance of the proposed GDM model in batch learning (section 4.2), incremental learning (section 4.3), and incremental learning with memory replay (section 4.4). We analyze and evaluate our model with the CORe50 dataset (Lomonaco and Maltoni, 2017; see section 4.1), a recently published benchmark for continuous object recognition from video sequences. We reproduce three experimental conditions defined by the CORe50 benchmark (section 4.5) showing that our model significantly outperforms state-of-the-art lifelong learning approaches. For the replication of these experiments, the source code of the GDM model is available as a repository[1].

[1]GDM model: https://github.com/giparisi/GDM

4.1. Feature Extraction

The CORe50 comprises 50 objects within 10 categories with image sequences captured under different conditions and multiple views of the same objects (varying background, object pose, and degree of occlusion; see **Figure 2**). Each object comprises a video sequence of approximately 15 s where the object is shown to the vision sensor held by a human operator. The video sequences were collected in 11 sessions (8 indoors, 3 outdoors) with a Kinect 2.0 sensor delivering RGB (1027 × 575) and depth images (512 × 242) at 20 frames per second (fps) for a total of 164,866 frames. For our experiments, we used 128 × 128 RGB images provided by the dataset at a reduced frame rate of 5hz. The movements performed by the human operator with the objects (e.g., rotation) are quite smooth and reducing the number of frames per second has not shown significant loss of information.

For a more direct comparison with the baseline results provided by Lomonaco and Maltoni (2017) who adopted the VGG model (Simonyan and Zisserman, 2014) pre-trained on the ILSVRC-2012 dataset (Russakovsky et al., 2014), our feature extraction module consists of the same pre-trained VGG model to which we applied a convolutional operation with 256 1 × 1 kernels on the output of the fully-connected hidden layer to reduce its dimensionality from 2048 to 256. Therefore, G-EM receives a 256-dimensional feature vector per sequence frame. Such compression of the feature vectors is desirable since the Gamma-GWR uses the Euclidean distance as a metric to compute the BMUs, which becomes weakly discriminant when the data are very high-dimensional or sparse (Parisi et al., 2015). Furthermore, it is expected that different pre-trained models may exhibit a slightly better performance than VGG, e.g., ResNet-50 (He et al., 2015; see Lomonaco and Maltoni, 2018 for ResNet-50 performance on CORe50). However, here we focus on showing the contribution of context-aware growing networks rather than comparing deep feature extractors.

4.2. Batch Learning

In the batch learning strategy, we trained the architecture on the entire training data at once and subsequently tested its classification performance at instance and category level. Following the same evaluation scheme described by Lomonaco and Maltoni (2017), we used the samples from sessions #3, #7, #10 for testing and the samples from the remaining 8 sessions for training. We compare our results to the baseline provided by Lomonaco and Maltoni (2017) using fine-tuning on a pre-trained VGG network (VGG+FT). To better assess the contribution of the temporal context (TC) for the task of continuous object recognition, we performed batch learning experiments with 3 different model configurations:

- **GDM**: We trained the model using TC and tested it on novel sequences. For each input frame, an object instance and an object category are predicted.
- **GDM (No TC)**: We trained and tested the model without TC by setting $K = 0$, i.e., the computation of the BMU is reduced to $b = \arg\min_{j \in A} \|\mathbf{x}(t) - \mathbf{w}_j\|^2$.

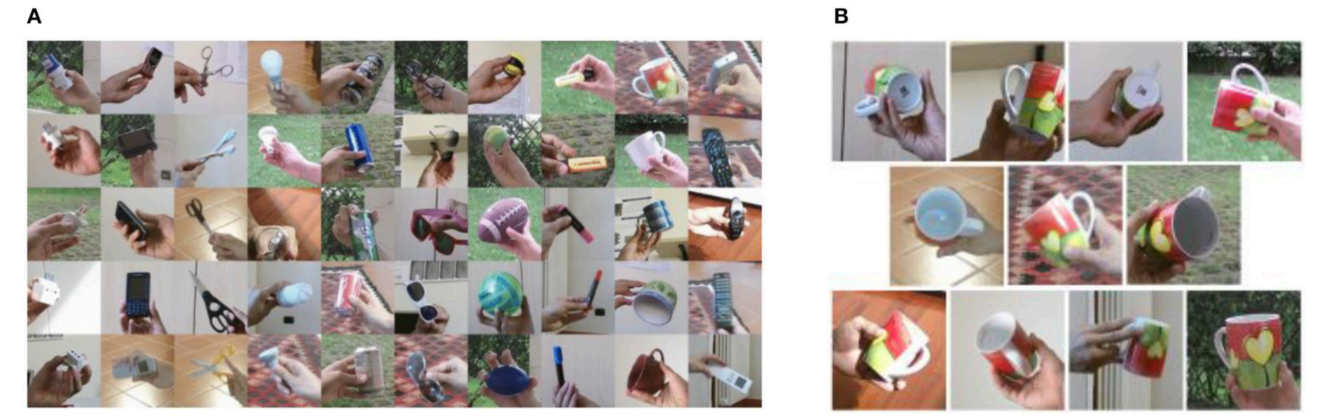


FIGURE 2 | The CORe50 dataset designed for continuous object recognition: **(A)** Example frames of the 10 categories (columns) comprising 5 object instances each, **(B)** Example frames for one object instance from the 11 acquisition sessions showing different background, illumination, pose, and degree of occlusion. Adapted from Lomonaco and Maltoni (2017).

- **GDM (No TC during test)**: We trained the model with TC but tested on single image frames by setting $K = 0$.

The training hyperparameters are listed in **Table 1**. Except for the insertion thresholds a_T^{EM} and a_T^{SM}, the remaining parameters were set similar to Parisi et al. (2017) for the incremental learning of sequences. Larger insertion thresholds will lead to a larger number of neurons. However, the best classification performance will not be necessarily obtained by the largest number of neurons. In G-EM, the neural representation should be characterized by a sufficiently high spatiotemporal resolution for discriminating between similar object instances and replaying episodic experience in the absence of sensory input. Conversely, regulated unsupervised learning in G-SM will lead to a more compact, overlapping neural representation with a smaller number of neurons while preserving the ability to correctly classify its input. The number of context descriptors (K^{EM}, K^{SM}) is set to 2. This means that G-EM neurons will activate in correspondence of 3 image frames and G-SM neurons in correspondence of 3 G-EM neurons, i.e., a processing window of 5 frames (1s of video at 5fps). Additional experiments showed that increasing the number of context descriptors does not significantly improve the overall accuracy. This is because a small number of context descriptors will lead to learning short-term temporal relations which are useful for temporal slowness, i.e., neurons that activate for multiple similar views of the same object (where different views of the object are induced by object motion). Neurons with a higher temporal depth will learn longer-term temporal relations and, depending on the difference between the training and test set, training with longer sequences may result in the specialization of neurons to the sequences in the training set while failing to generalize. Therefore, convenient values for K^{EM} and K^{SM} can be selected according to different criteria and properties of the input, e.g., number of frames per second, smoothness of object motion, desired degree of neural specialization.

TABLE 1 | Training hyperparameters for the G-EM and G-SM networks (batch and incremental learning).

Hyperparameters	**Value**
Insertion thresholds	$a_T^{\mathrm{EM}} = 0.3$, $a_T^{\mathrm{SM}} = 0.001$
Habituation counters	$h_T = 0.1$, $\tau_b = 0.3$, $\tau_n = 0.1$, $\kappa = 1.05$
Temporal depth	$K^{\mathrm{EM}} = 2$, $K^{\mathrm{SM}} = 2$
Temporal context	$\alpha = [0.67, 0.24, 0.09]$, $\beta = 0.7$
Learning rates	$\epsilon_b = 0.5$, $\epsilon_n = 0.005$

TABLE 2 | Comparison of batch learning performance for instance-level and category-level classification.

Approach	**Accuracy (%) (Instances)**	**Accuracy (%) (Categories)**
VGG + FT (Lomonaco and Maltoni, 2017)	69.08	80.23
Proposed GDM (No TC)	70.42	83.54
Proposed GDM (No TC during test)	72.56	87.32
Proposed GDM	**79.43**	**93.92**

We show the accuracy for the pre-trained VGG with fine-tuning (VGG+FT) and the proposed GDM for three different configurations: (i) growing networks with temporal context (TC), (ii) without TC, and (iii) without TC during test. Best results in bold.

The classification performance for the 3 different configurations is summarized in **Table 2**, showing instance-level and category-level accuracy after 35 training epochs averaged across 5 learning trials in which we randomly shuffled the batches from different sessions. The best results were obtained by GDM using temporal context with an average accuracy of 79.43% (instance level) and 93.92% (category level), showing an improvement of 10.35 and 13.69%, respectively, with respect to the baseline results (Lomonaco and Maltoni, 2017). Without the use of temporal context, the accuracy is comparable to the baseline showing a marginal improvement of 1.34% (instance level) and 3.31% (category level). Our results

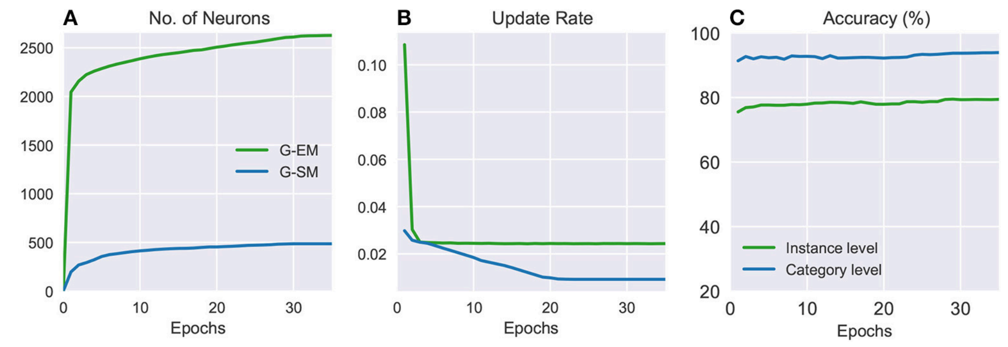


FIGURE 3 | Batch learning on the CORe50: numbers of neurons **(A)**, update rates **(B)**, and classification accuracies **(C)** of G-EM and G-SM through 35 training epochs averaged across 5 learning trials.

demonstrate that learning the temporal relations of the input plays an important role for this dataset. Interestingly, dropping the temporal component during the test phase, i.e., using single image frames for testing on context-aware networks, shows a slightly better performance (2.14 and 3.78%, respectively) than training without temporal context. This is because trained neural weights embed some temporal structure of the training sequences and, consequently, the context-free computation of a BMU from a single input frame will still be matched to context-aware neurons.

Figure 3 shows the number of neurons, update rate, and classification accuracy for G-EM and G-SM (with temporal context) through 35 training epochs averaged across 5 learning trials. It can be seen that the average number of neurons created in G-EM is significantly higher than in G-SM (**Figure 3A**). This is expected since G-EM will grow to minimize the discrepancy between the input and its internal representation, whereas neurogenesis and neural update rate in G-SM are regulated by the ability of the network to predict the correct class labels of the input. The update rate (**Figure 3B**) is given by multiplying the fixed learning rate by the habituation counter of the neurons ($\epsilon_i \cdot h_i$), which shows a monotonically decreasing behavior. This indicates that, after a number of epochs, the created neurons become habituated to the input. Such a habituation mechanism has the advantage of protecting consolidated knowledge from being disrupted or overwritten by the learning of novel sensory experience, i.e., well-trained neurons will respond slower to changes in the distribution and the network will create new neurons to compensate for the discrepancy between the input and its representation.

4.3. Incremental Learning

In the incremental learning strategy, the training samples of different object categories become progressively available over time, i.e., each mini-batch contains all the instances of an object category from all the 8 training sessions. Each category batch is shown once to the model and samples from that category are not shown again during the learning of new categories. Therefore, the model must incrementally learn new object instances and categories without forgetting previously learned ones. For a direct comparison with our previous experiment, the hyperparameters for the incremental learning experiment are the same as for the batch learning strategy (**Table 1**).

Figure 4 shows the number of neurons, update rate, and accuracy over 10 epochs (i.e., the 10 object categories) averaged across 5 runs of randomly shuffled object categories. The variance from the mean values (shaded areas in **Figure 4**) shows that the order of exposure to object categories can affect the final result. In general, the number of neurons increases over time (**Figure 4A**) and, in contrast to the batch learning strategy where neurogenesis is particularly strong during the initial training epochs (**Figure 3A**), in this case new neurons are progressively created in response to the exposure of the model to novel object classes. Similarly, the update rate for both networks (**Figure 4B**) does not monotonically decrease over time but rather stays quite stable in correspondence to novel sensory experience. Since newly created neurons are not well trained, the update rate will be higher at the moment of neural insertion and progressively decrease as the newly created neurons become habituated. The overall accuracy decreases with the number of object categories encountered, showing a higher sensitivity of the model with respect to the order in which the object categories are presented (**Figure 4C**). The average classification accuracy for the incremental learning strategy is 75.93% $\pm$ 2.23 (instance level) and 85.53% $\pm$ 1.35 (category level), showing a decrease of 3.5% and 8.39%, respectively, compared to the batch learning performance. This suggests that an additional mechanism such as memory replay is required to prevent the disruptive interference of existing representations.

Figure 5 shows a comparison of the effects of forgetting during the incremental learning strategy in terms of the overall accuracy on the categories encountered so far and the accuracy on the first encountered category as new categories are learned. For the object instances, we compare the overall accuracy (**Figure 5A**) with the accuracy on the first 5 encountered instances (i.e., 1 category; see **Figure 5B**), showing that for the latter the accuracy drops to 69.25%$\pm$4.31 (compared to 75.93%$\pm$ 2.23). For the object categories (**Figures 5C,D**), the accuracy on the first encountered category drops to 79.53% $\pm$ 5.23 (compared to 85.53% $\pm$ 1.35). Overall, these results suggest that memory

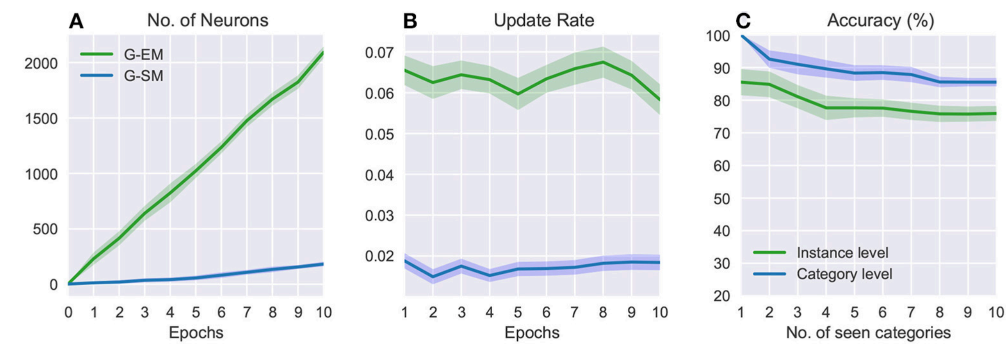


FIGURE 4 | Incremental learning: numbers of neurons **(A)**, update rates **(B)**, and classification accuracies **(C)** over 10 categories averaged across 5 learning trials. The shaded areas show the standard deviation.

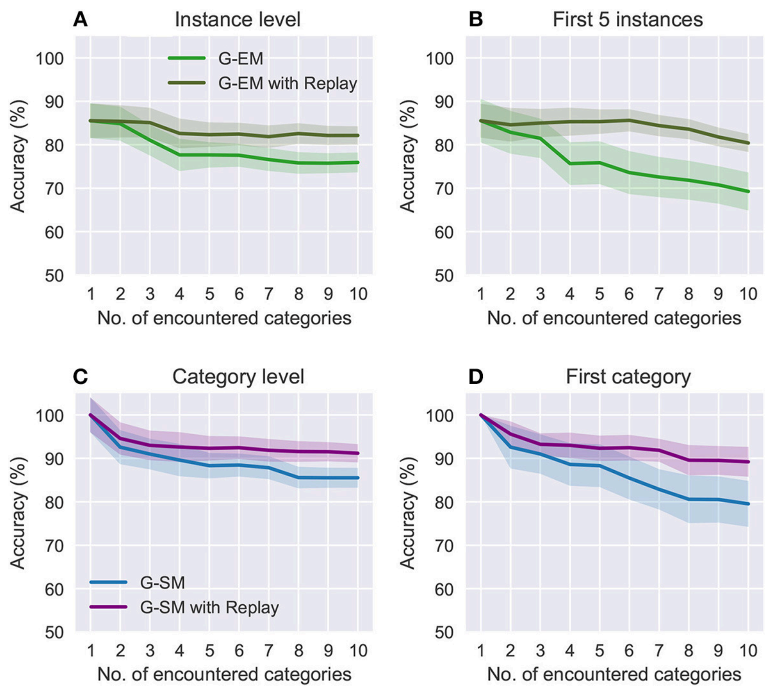


FIGURE 5 | Comparison of the effects of forgetting during incremental learning with and without memory replay at an instance level **(A,B)** and category level **(C,D)**. Each category contains 5 instances. The plots show the average accuracies on the categories encountered so far **(A,C)** and the accuracies on the first encountered category **(B,D)** as further new categories are learned. The shaded areas show the standard deviation.

replay is an important feature for the reactivation of previously learned neural representations at the moment of learning from novel sensory experience with the goal to prevent that classes that have been encountered at early stages be forgotten over time.

4.4. Incremental Learning With Memory Replay

In this learning strategy, we trained the model as described above with progressively available mini-batches containing 1 object category each. Here, however, after each learning episode (i.e., a training epoch over the mini-batch), the model generates a set of RNATs, S_j (Equations 10 and 11) from the G-EM neurons. Thus, the number of RNATs of length $\lambda = 5$ is equal to the number of neurons created by G-EM. The set of RNATs is replayed to G-EM and G-SM in correspondence of novel sensory experience to reinforce previously encountered categories. Since the growing self-organizing networks store the global temporal context, $\mathbf{C}_k(t)$, over the training iterations (Equation 3) for learning the temporal structure of the input, each RNAT is fed into G-EM and G-SM as a single sample batch and the global temporal context is reset to zero after one epoch. It is expected that, by periodically replaying RNATs when new categories are encountered, knowledge representations will consolidate over time and, consequently, significantly alleviate catastrophic forgetting for sustained learning periods.

The benefit of using memory replay is shown in **Figure 5** where we compare the overall accuracy on all the categories encountered so far to the accuracy on the first encountered category over the number of encountered categories. At an instance level (**Figures 5A,B**), incremental learning with memory replay improves the overall accuracy to 82.14% ± 2.05 (from 75.93% ± 2.23) and accuracy on the first 5 instances to 80.41% ± 1.35 (from 69.25% ± 2.01). At a category level (**Figures 5C,D**), the overall accuracy increases to 91.18% ± 0.25 (from 85.53% ± 1.35) and the accuracy on the first encountered category to 89.21% ± 3.37 (from 79.53% ± 5.23) Overall, our results support the hypothesis that replaying RNATs generated from G-EM mitigates the effects of catastrophic forgetting.

4.5. Continuous Object Recognition

We evaluate our model with the 3 incremental learning scenarios proposed by the CORe50 benchmark for the task of continuous object recognition:

New Instances (NI): New instances of the same class and from different acquisition sessions become progressively available and are shown once to the model. Therefore, all the classes to be learned are known. For all the classes, the model is trained with the instances of a first session and subsequently with the remaining 7 sessions. (Here, the term *classes* is used for object categories.)

New Classes (NC): Training samples from novel different classes become available over time, thus the model must deal with the learning of new classes without forgetting previously learned ones. Each training batch contains all the sequences of a small group of classes and memory replay is possible across batches. The first batch includes 10 objects while the remaining 8 batches contain 5 objects each. The test set includes samples from all the classes and the model is required to classify samples that have not been seen yet (except for the last evaluation step).

New Instances and Classes (NIC): New instances and classes become available over time, requiring the model to consolidate knowledge about known classes and to learn new ones. The first batch includes 10 classes and the subsequent batches 5 classes each, with only one training sequence per class included in the batches. This scenario comprises 79 batches, maximizing the categorical representation in the first batch and randomly selecting the remaining 78 batches.

For each scenario, we compute the average accuracy over 10 configurations of randomly shuffled batches. The results for the NI, NC, and NIC scenarios compared to other approaches are listed in **Table 3**. It can be seen that our proposed method with memory replay produces state-of-the-art results for this benchmark dataset, showing an average accuracy of 87.94, 86.14, and 87.06% for the NI, NC, and NIC scenarios, respectively. These results represent a large increase in accuracy over 20% for each scenario with respect to the previous best results, i.e., a cumulative approach reported by Lomonaco and Maltoni (2017). The authors reported results using 3 methods with pre-trained CNN models and 128 × 128 images: (i) a naïve approach which consists of continuous stochastic gradient descent training as new batches become available, (ii) a proposed *CopyWeights with Re-init* (CWR) method that skips layers *fc6* and *fc7* of the CNN (for details, see Lomonaco and Maltoni, 2017; page 7), and (iii) a cumulative approach where the learning is carried out by considering the current batch and all the previous ones.

TABLE 3 | Accuracy on the CORe50 incremental learning scenarios.

Method	Avg. Acc. (%)	Std. Dev. (%)
NEW INSTANCES (NI)		
Proposed GDM (with replay)	**87.94**	1.72
Proposed GDM	74.87	2.54
Cumulative (Lomonaco and Maltoni, 2017)	65.15	0.66
LwF (Li and Hoiem, 2016)	59.42*	2.71
EWC (Kirkpatrick et al., 2017)	57.40*	3.80
Naïve (Lomonaco and Maltoni, 2017)	54.69	6.18
NEW CLASSES (NC)		
Proposed GDM (with replay)	**86.14**	2.03
Proposed GDM	73.02	2.91
Cumulative	64.65	1.04
iCaRL (Rebuffi et al., 2017)	43.62*	0.66
CWR (Lomonaco and Maltoni, 2017)	42.32	1.09
LwF	27.60*	1.70
EWC	26.22*	1.18
Naïve	10.75	0.84
NEW INSTANCES AND CLASSES (NIC)		
Proposed GDM (with replay)	**87.06**	2.13
Proposed GDM	72.57	2.96
Cumulative	64.13	0.88
CWR	29.56	0
LwF	28.94*	4.30
EWC	28.31*	4.30
Naïve	19.39	2.90

*Results denoted with * indicate the re-implementation of the method by Lomonaco and Maltoni (2017). Best results in bold.*

Ours and previously reported experiments show that lifelong learning is a very challenging task and that the overall performance of some approaches can differ significantly according to the specific learning strategy. Furthermore, a more direct comparison of the model's behavior is hindered by the fact that the other methods do not comprise recurrent neural dynamics that account for learning the temporal structure of the input which, in this case, is a clear advantage (see **Table 2**) since the temporal relations of the input can be exploited for more robust learning and prediction.

The experiments reported for all the 3 incremental learning scenarios were conducted with the test set containing samples from all the seen classes (except for the last evaluation step). Such an evaluation scheme was selected to keep the test set consistent across all the scenarios (Lomonaco and Maltoni, 2017). However, in a more realistic lifelong learning scenario, the model should be able to deal with unknown classes during sequence retrieval. In our case, the model will always predict an output label in correspondence to a retrieved sequence. Instead, it would be convenient to design a novelty detection mechanism for unseen classes so that the system will output a predicted label provided that the input sequence produces a sufficient level of neural activity (Marsland et al., 2002; Parisi et al., 2015).

5. DISCUSSION

5.1. Summary

We proposed a growing dual-memory architecture with self-organizing networks for the lifelong learning of spatiotemporal representations from image sequences. The GDM model can perform continuous object recognition at an instance level (episodic experience) and at a category level (semantic knowledge). We introduced the use of recurrent self-organizing networks, in particular of extended versions of the Gamma-GWR, to model the interplay of two complementary learning systems: an episodic memory, G-EM, with the task of learning fine-grained spatiotemporal representations from sensory experience and a semantic memory, G-SM, for learning more compact representations from episodic experience. With respect to previously proposed dual-memory learning systems, our contribution is threefold. First, in contrast to the predominant approach of processing static images independently, we implement recurrent self-organizing memories for learning the spatiotemporal structure of the input. Second, as a complementary mechanism to unsupervised growing networks, we use task-relevant signals to regulate structural plasticity levels in the semantic memory, leading to the development of more compact representations from episodic experience. Third, we model memory replay as the periodic reactivation of neural activity trajectories from temporal synaptic patterns embedded in an episodic memory. Our experiments show that the proposed GDM model significantly outperforms state-of-the-art lifelong learning methods in three different incremental learning tasks with the CORe50 benchmark dataset.

5.2. Growing Recurrent Networks With Memory Replay

The use of growing networks leads to the dynamic allocation of additional neurons and connections in response to novel sensory experience. In particular, the Gamma-GWR (Parisi et al., 2017) provides the basic mechanism for growing self-organizing memories with temporal context for learning the spatiotemporal structure of the input in an unsupervised fashion. Different models of neural network self-organization have been proposed that resemble the dynamics of Hebbian learning and plasticity (Fritzke, 1995; Kohonen, 1995; Marsland et al., 2002), with neural map organization resulting from unsupervised statistical learning. For instance, in the traditional self-organizing feature map and its dynamic variant (e.g., Kohonen, 1995; Rougier and Boniface, 2011, the number of neurons is pre-defined. Empirically selecting a convenient number of neurons can be tedious for networks with recurrent dynamics, especially when dealing with non-stationary input distributions (Strickert and Hammer, 2005). To alleviate this issue, growing self-organizing networks for temporal processing have been proposed, for instance the Gamma-GNG (Estévez and Vergara, 2012) that equips neurons with a temporal context. However, the Gamma-GNG grows at a constant, pre-defined interval and does not consider whether previously created neurons have been well trained before creating new ones. This will lead to scalability issues if the selected interval is too short or, conversely, to an insufficient number of neurons if the interval is too large. Therefore, we extended the Gamma-GWR which can quickly react to changes in the input distribution and can create new neurons whenever they are required.

From a biological perspective, there has been controversy over whether in human adults detectable amounts of new neurons can grow. Recent research has suggested that hippocampal neurogenesis drops sharply in children (Sorrells et al., 2018) and becomes undetectable in adulthood, whereas other studies suggest that hippocampal neurogenesis sustains human-specific cognitive function throughout life (Boldrini et al., 2018). Neurophysiological studies evidence that, in addition to neurogenesis, synaptic rewiring by structural plasticity has a significant contribution on memory formation in adults (Knoblauch et al., 2014; Knoblauch, 2017), with a major role of structural plasticity in increasing information storage efficiency in terms of space and energy demands. While the mechanisms for creating new neurons and connections in the Gamma-GWR do not resemble biologically plausible mechanisms of neurogenesis and synaptogenesis (e.g., Eriksson et al., 1998; Ming and Song, 2011; Knoblauch, 2017), the GWR learning algorithm represents an efficient computational model that incrementally adapts to non-stationary input. Crucially, the GWR model creates new neurons whenever they are required and only after the training of existing ones. The neural update rate decreases as the neurons become more habituated, which has the effect of preventing that noisy input interferes with consolidated neural representations. Alternative theories suggest that an additional function of hippocampal neurogenesis is the encoding of time for the formation of temporal associations in memory (Aimone et al., 2006, 2009), e.g., in terms of temporal clusters of long-term episodic memories. This represents an interesting research direction for the modeling of temporal associations in the Gamma-GWR.

For mitigating catastrophic forgetting during incremental learning tasks, the proposed model generates recurrent neural activity trajectories (RNATs; Equation 10) after each learning episode. The set of generated RNATs is periodically replayed to both networks in correspondence of novel sensory experience for the consolidation of knowledge over time. This is consistent with biological evidence suggesting that the reactivation of hippocampal representations and their frequent replay to the neocortex are crucial for memory consolidation and retrieval (see Carr et al., 2011 for a review). The process of replaying previously seen data without explicitly storing data samples is referred to as intrinsic replay (Robins, 1995) and has the advantage of fewer memory requirements with respect to explicitly storing training samples. In our approach, the episodic memory G-EM embeds the temporal structure of the input through the implementation of temporal synapses that are strengthened between consecutively activated neurons (Equation 9). Therefore, RNATs comprise prototype sequence snapshots that can be generated without the need of explicitly storing the training sequences. Our reported results show that the use of RNATs yields a significantly improved overall accuracy during incremental learning.

In this work, we have focused on regulating the mechanisms of neurogenesis and neural update, whereas we have not investigated the removal of old connections and isolated neurons. At each learning iteration of the Gamma-GWR, old connections exceeding a given age threshold and neurons without connections can be deleted. Removing a neuron from the network means that the knowledge coded by that unit is permanently forgotten. Therefore, a convenient maximum age of connections must be set to avoid catastrophic forgetting. In incremental learning scenarios, it is non-trivial to define a convenient age threshold for connections to be removed since data samples become available over time and neurons coding for consolidated knowledge might not fire for a large number of iterations. Mechanisms of intrinsic memory replay as modeled in this paper could be used to prevent the deletion of consolidated knowledge. For instance, the periodic replay of episodic representations would prevent the networks from deleting relevant knowledge also when external sensory input does not activate those representations for sustained periods of time.

Conceptual similarities can be found between our model and the adaptive resonance theory (ART) in which neurons are iteratively adapted to a non-stationary input distribution in an unsupervised fashion and new neurons can be added in correspondence of dissimilar input (see Grossberg, 2012 for a review). The primary intuition of the ART model is that learning occurs via the interaction of top-down and bottom-up processes, where top-down expectations act as memory templates or prototypes which are compared to bottom-up sensory observations. Similar to the GWR's activation threshold, the ART model uses a vigilance parameter to produce fine-grained or more general memories. Despite its inherent ability to mitigate catastrophic forgetting during incremental learning, it has been noted that the results of some variants of the ART model depend significantly upon the order in which the training data are processed. However, an extensive evaluation with recent lifelong learning benchmarks has not been reported. Therefore, ART-based models represent an additional complementary approach to growing self-organizing models.

6. CONCLUSION

Lifelong learning represents a fundamental but challenging component of artificial systems and autonomous agents. Despite significant advances in this direction, current models of lifelong learning are far from providing the flexibility, robustness, and scalability exhibited by biological systems. In this paper, we contribute to extending dual-memory models for the processing of sequential input which represents more realistic experimental settings compared to learning from static image datasets. In the future, it would be interesting to extend this model to the multisensory domain, e.g., where neural representations can be continually learned from audio-visual streams (Parisi et al., 2016, 2018b). The proposed architecture can be considered a further step toward more flexible lifelong learning methods that can be deployed in embodied agents for incrementally acquiring and refining knowledge over sustained periods through the active interaction with the environment.

AUTHOR CONTRIBUTIONS

GP, JT, CW, and SW designed the experiments. GP wrote the paper. JT, CW, and SW revised the paper.

ACKNOWLEDGMENTS

This research was partially supported by the German Research Foundation (DFG) under project Transregio Crossmodal Learning (TRR 169). The authors would like to thank Pablo Barros and Vadym Gryshchuk for technical support.

REFERENCES

Aimone, J. B., Wiles, J., and Gage, F. H. (2006). Potential role for adult neurogenesis in the encoding of time in new memories. *Nat. Neurosci.* 9, 723–727. doi: 10.1038/nn1707

Aimone, J. B., Wiles, J., and Gage, F. H. (2009). Computational influence of adult neurogenesis on memory encoding. *Neuron* 61, 187–202. doi: 10.1016/j.neuron.2008.11.026

Benna, M. K., and Fusi, S. (2016). Computational principles of synaptic memory consolidation. *Nat. Neurosci.* 19, 1697–1708. doi: 10.1038/nn.4401

Berkes, P., and Wiskott, L. (2005). Slow feature analysis yields a rich repertoire of complex cell properties. *J. Vis.* 5, 579–602. doi: 10.1167/5.6.9

Boldrini, M., Fulmore, C. A., Tartt, A. N., Simeon, L. R., Pavlova, I., Poposka, V., et al. (2018). Human hippocampal neurogenesis persists throughout aging. *Cell Stem Cell* 22, 589–599. doi: 10.1016/j.stem.2018.03.015

Booth, M. C., and Rolls, E. T. (1998). View-invariant representations of familiar objects by neurons in the inferior temporal visual cortex. *Cereb. Cortex* 8, 510–523. doi: 10.1093/cercor/8.6.510

Cangelosi, A., and Schlesinger, M. (2015). *Developmental Robotics: From Babies to Robots*. MIT Press.

Carr, M. F., Jadhav, S. P., and Frank, L. M. (2011). Hippocampal replay in the awake state: a potential substrate for memory consolidation and retrieval. *Nat. Neurosci.* 14, 147–153. doi: 10.1038/nn.2732

Clarke, A., and Tyler, L. K. (2014). Object-specific semantic coding in human perirhinal cortex. *J. Neurosci.* 34, 4766–4775. doi: 10.1523/JNEUROSCI.2828-13.2014

Ditzler, G., Roveri, M., Alippi, C., and Polikar, R. (2015). Learning in nonstationary environments: a survey. *IEEE Comput. Intell. Mag.* 10, 12–25. doi: 10.1109/MCI.2015.2471196

Draelos, T. J., Miner, N. E., Lamb, C. C., Vineyard, C. M., Carlson, K. D., James, C. D., et al. (2017). "Neurogenesis deep learning," in *IJCNN'17* (Anchorage, AK), 526–533.

Einhäuser, W., Hipp, J., Eggert, J., Körner, E., and König, P. (2005). Learning viewpoint invariant object representations using a temporal coherence principle. *Biol. Cybern.* 93, 79–90. doi: 10.1007/s00422-005-0585-8

Eriksson, P. S., Perfilieva, E., Bjork-Eriksson, T., Alborn, A. M., Nordborg, C., Peterson D. A., et al. (1998). Neurogenesis in the adult human hippocampus. *Nat. Med.* 4, 1313–1317. doi: 10.1038/3305

Estévez, P. A., and Vergara, J. R. (2012). "Nonlinear time series analysis by using gamma growing neural gas," in *Workshop on Self-Organizing Maps (WSOM)* (Santiago), 205–214.

French, R. M. (1997). Pseudo-recurrent connectionist networks: an approach to the sensitivity-stability dilemma. *Connect. Sci.* 9, 353–380. doi: 10.1080/095400997116595

Fritzke, B. (1995). "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7* (Denver, CO: MIT Press), 625–632.

Gelbard-Sagiv, H., Mukamel, R., Harel, M., Malach, R., and Fried, I. (2008). Internally generated reactivation of single neurons in human hippocampus during free recall. *Science* 322, 96–101. doi: 10.1126/science.1164685

Gepperth, A., and Karaoguz, C. (2015). A bio-inspired incremental learning architecture for applied perceptual problems. *Cogn. Comput.* 8, 924–934. doi: 10.1007/s12559-016-9389-5

Grossberg, S. (1980). How does a brain build a cognitive code? *Psychol. Rev.* 87, 1–51. doi: 10.1037/0033-295X.87.1.1

Grossberg, S. (2007). Consciousness CLEARS the mind. *Neural Netw.* 20, 1040–1053. doi: 10.1016/j.neunet.2007.09.014

Grossberg, S. (2012). Adaptive Resonance Theory: how a brain learns to consciously attend, learn, and recognize a changing world. *Neural Netw.* 37, 1–47. doi: 10.1016/j.neunet.2012.09.017

He, K., Zhang, X., Ren, S., Sun, J. (2015). Deep residual learning for image recognition. *arXiv:1512.03385*.

Kamra, N., Gupta, U., and Liu, Y. (2018). Deep generative dual memory network for continual learning. *arXiv [Preprint]*:1710.10368.

Karimi-Rouzbahani, H., Bagheri, N., and Ebrahimpour, R. (2017). Invariant object recognition is a personalized selection of invariant features in humans, not simply explained by hierarchical feed-forward vision models. *Sci. Rep.* 7:14402. doi: 10.1038/s41598-017-13756-8

Karlsson, M. P., and Frank, L. M. (2009). Awake replay of remote experiences in the hippocampus. *Nat. Neurosci.* 12, 913–918. doi: 10.1038/nn.2344

Kemker, R., and Kanan, C. (2018). "FearNet: brain-inspired model for incremental learning," in *ICLR'18* (Vancouver, BC).

Kemker, R., McClure, M., Abitino, A., Hayes, T., and Kanan, C. (2018). "Measuring catastrophic forgetting in neural networks," in *AAAI'18* (New Orleans, LA).

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. U.S.A.* 114, 3521–3526. doi: 10.1073/pnas.1611835114

Kitamura, T., Ogawa, S. K., Roy, D. S., Okuyama, T., Morrissey, M. D., Smith, L. M. et al. (2017). Engrams and circuits crucial for systems consolidation of a memory. *Science* 356, 73–78. doi: 10.1126/science.aam6808

Knoblauch, A. (2017). "Impact of structural plasticity on memory formation and decline," in *Rewiring the Brain: A Computational Approach to Structural Plasticity in the Adult Brain*, Vol. 17, eds A. van Ooyen and M. Butz (London: Elsevier, Academic Press), 361–386.

Knoblauch, A., Körner, E., Körner, U., and Sommer, F. T. (2014). Structural plasticity has high memory capacity and can explain graded amnesia, catastrophic forgetting, and the spacing effect. *PLoS ONE* 9:e96485. doi: 10.1371/journal.pone.0096485

Kohonen, T. (1995). *Self-Organizing Maps*. New York, NY: Springer.

Krizhevsky, A. (2009). *Learning Multiple Layers of Features from Tiny Images*, Master's thesis, University of Toronto.

Kudrimoti, H. S., Barnes, C. A., and McNaughton, B. L. (1999). Reactivation of hippocampal cell assemblies: effects of behavioral state, experience, and EEG dynamics. *J. Neurosci.* 19, 4090–4101. doi: 10.1523/JNEUROSCI.19-10-04090.1999

Kumaran, D., Hassabis, D., and McClelland, J. L. (2016). What learning systems do intelligent agents need? Complementary learning systems theory updated. *Trends Cogn. Sci.* 20, 512–534. doi: 10.1016/j.tics.2016.05.004

Kumaran, D., and McClelland, J. L. (2012). Generalization through the recurrent interaction of episodic memories: a model of the hippocampal system. *Psychol. Rev.* 119, 573–616. doi: 10.1037/a0028681

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521, 436–444. doi: 10.1038/nature14539

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 2278–2324. doi: 10.1109/5.726791

Lewkowicz, D. J. (2014). Early experience & multisensory perceptual narrowing. *Dev. Psychobiol.* 56, 292–315. doi: 10.1002/dev.21197

Li, Z. and Hoiem, D. (2016). "Learning without forgetting," in *European Conference on Computer Vision* (Amsterdam: Springer), 614–629.

Lomonaco, V., and Maltoni, D. (2017). "CORe50: a new dataset and benchmark for continuous object recognition," in *CoRL'17* (Mountain View, CA).

Lomonaco, V., and Maltoni, D. (2018). Continuous learning in single-incremental-task scenarios. *arXiv [Preprint]*:1806.08568.

Marsland, S., Shapiro, J., and Nehmzow, U. (2002). A self-organising network that grows when required. *Neural Netw.* 15, 1041–1058. doi: 10.1016/S0893-6080(02)00078-3

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychol. Rev.* 102, 419–457. doi: 10.1037/0033-295X.102.3.419

McCloskey, M., and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: the sequential learning problem. *Psychol. Learn. Motiv.* 24, 104–169. doi: 10.1016/S0079-7421(08)60536-8

Mermillod, M., Bugaiska, A. and Bonin, P. (2013). The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Front. Psychol.* 4:504. doi: 10.3389/fpsyg.2013.00504

Mineiro, P., and Zipser, D. (1998). Analysis of direction selectivity arising from recurrent cortical interactions. *Neural Comput.* 10, 353–371. doi: 10.1162/089976698300017791

Ming, G. L., and Song, H. (2011). Adult neurogenesis in the mammalian brain: Significant answers and significant questions. *Neuron* 70, 687–702. doi: 10.1016/j.neuron.2011.05.001

Neunuebel, J. P., and Knierim, J. J. (2014). CA3 retrieves coherent representations from degraded input: direct evidence for CA3 pattern completion and dentate gyrus pattern separation. *Neuron* 81, 416–427. doi: 10.1016/j.neuron.2013.11.017

O'Reilly, R. C., and Norman, K. A. (2002). Hippocampal and neocortical contributions to memory: advances in the complementary learning systems framework. *Trends Cogn. Sci.* 6, 505–5010. doi: 10.1016/S1364-6613(02)02005-3

Parisi, G. I., Barros, P., Fu, D., Magg, S., Wu, H., Liu, X., et al. (2018b). A neurorobotic experiment for crossmodal conflict resolution in complex environments. *arXiv [Preprint]*:1802.10408.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2018a). Continual lifelong learning with neural networks: a review. *arXiv [Preprint]*:1802.07569.

Parisi, G. I., Tani, J., Weber, C., and Wermter, S. (2016). Emergence of multimodal action representations from neural network self-organization. *Cogn. Syst. Res.* 43, 208–221. doi: 10.1016/j.cogsys.2016.08.002

Parisi, G. I., Tani, J., Weber, C., and Wermter, S. (2017). Lifelong learning of humans actions with deep neural network self-organization. *Neural Netw.* 96, 137–149. doi: 10.1016/j.neunet.2017.09.001

Parisi, G. I., Weber, C., and Wermter, S. (2015). Self-organizing neural integration of pose-motion features for human action recognition. *Front. Neurorobot.* 9:3. doi: 10.3389/fnbot.2015.00003

Power, J. D., and Schlaggar, B. L. (2016). Neural plasticity across the Lifespan. *Wiley Interdiscip. Rev.* 6:216. doi: 10.1002/wdev.216

Principe, J., Kuo, J. M., and Celebi, S. (1994). An analysis of the gamma memory in dynamic neural models. *IEEE Trans. Neural Netw.* 5, 331–337. doi: 10.1109/72.279195

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). iCaRL: incremental classifier and representation learning. *arXiv:1611.07725*, 2001–2010.

Robins, A. V. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connect. Sci.* 7, 123–146. doi: 10.1080/09540099550039318

Rougier, N. P., and Boniface, Y. (2011). Dynamic self-organising map. *Neurocomputing* 74, 1840–1847. doi: 10.1016/j.neucom.2010.06.034

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2014). ImageNet large scale visual recognition challenge. *arXiv [Preprint]*:1409.0575.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., et al. (2016). Progressive neural networks. arXiv*[Preprint]*:1606.04671.

Simonyan, K., and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv [Preprint]*:1409.1556.

Sorrells, S. F., Paredes, M. F., Cebrian-Silla, A., Sandoval, K., Qi, D., Kelley, K. W., et al. (2018). Human hippocampal neurogenesis drops sharply in children to undetectable levels in adults. *Nature* 555, 377–381. doi: 10.1038/nature25975

Stanley, J. C. (1976). Computer simulation of a model of habituation. *Nature* 261, 146–148.

Strickert, M., and Hammer, B. (2005). Merge SOM for temporal data. *Neurocomputing* 64, 39–71. doi: 10.1016/j.neucom.2004.11.014

Tani, J. (2016). *Exploring Robotic Minds: Actions, Symbols, and Consciousness a Self-Organizing Dynamic Phenomena*. Oxford University Press.

Thrun, S., and Mitchell, T. M. (1995). Lifelong robot learning. *Robot. Auton. Syst.* 15, 25–46. doi: 10.1016/0921-8890(95)00004-Y

Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. (2018). Predicting higher visual cortex neural responses. *Proc. Natl. Acad. Sci. U.S.A.* 111, 8619–8624. doi: 10.1073/pnas.1403112111

Yassa, M. A., and Stark, C. E. (2011). Pattern separation in the hippocampus. *Trends Neurosci.* 34, 515–525. doi: 10.1016/j.tins.2011.06.006

Zenke, F., Gerstner, W., and Ganguli, S. (2017a). The temporal paradox of Hebbian learning and homeostatic plasticity. *Neurobiology* 43, 166–176. doi: 10.1016/j.conb.2017.03.015

Zenke, F., Poole, B., and Ganguli, S. (2017b). "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning, ICML'17* (Sydney, NSW), 3987–3995.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 14 December 2018
doi: 10.3389/fnbot.2018.00076

Active Fovea-Based Vision Through Computationally-Effective Model-Based Prediction

*Emmanuel Daucé**

Ecole Centrale de Marseille, INSERM, Institut de Neurosciences des Systèmes, Aix Marseille Université, Marseille, France

What motivates an action in the absence of a definite reward? Taking the case of visuomotor control, we consider a minimal control problem that is how select the next saccade, in a sequence of discrete eye movements, when the final objective is to better interpret the current visual scene. The visual scene is modeled here as a partially-observed environment, with a generative model explaining how the visual data is shaped by action. This allows to interpret different action selection metrics proposed in the literature, including the Salience, the Infomax and the Variational Free Energy, under a single information theoretic construct, namely the view-based Information Gain. Pursuing this analytic track, two original action selection metrics named the Information Gain Lower Bound (IGLB) and the Information Gain Upper Bound (IGUB) are then proposed. Showing either a conservative or an optimistic bias regarding the Information Gain, they strongly simplify its calculation. An original fovea-based visual scene decoding setup is then proposed, with numerical experiments highlighting different facets of artificial fovea-based vision. A first and principal result is that state-of-the-art recognition rates are obtained with fovea-based saccadic exploration, using less than 10% of the original image's data. Those satisfactory results illustrate the advantage of mixing predictive control with accurate state-of-the-art predictors, namely a deep neural network. A second result is the sub-optimality of some classical action-selection metrics widely used in the literature, that is not manifest with finely-tuned inference models, but becomes patent when coarse or faulty models are used. Last, a computationally-effective predictive model is developed using the IGLB objective, with pre-processed visual scan-path read-out from memory, bypassing computationally-demanding predictive calculations. This last simplified setting is shown effective in our case, showing both a competing accuracy and a good robustness to model flaws.

Keywords: intrinsic motivation, foveated vision, saccadic eye movements, active inference, information gain, convolutional neural networks (CNN), active vision

Edited by:
Vieri Giuliano Santucci, Istituto di Scienze e Tecnologie della Cognizione (ISTC), Italy

Reviewed by:
German I. Parisi, Universität Hamburg, Germany
Alex Pitti, Université de Cergy-Pontoise, France

Correspondence:
Emmanuel Daucé
edauce@ec-marseille.fr

Received: *31 May 2018*
Accepted: *05 November 2018*
Published: *14 December 2018*

Citation:
Daucé E (2018) Active Fovea-Based Vision Through Computationally-Effective Model-Based Prediction. Front. Neurorobot. 12:76. doi: 10.3389/fnbot.2018.00076

1. INTRODUCTION

In complement with goal-oriented activity, animal motor control also relates to the search for sensory cues in order to better interpret its sensory environment and improve action efficacy. This resorts to choosing relevant viewpoints, i.e., selecting body placement and/or sensors orientation in order to capture a sensory signal that should help disambiguate the current scene. The center of sight, in particular, is constantly and actively moving during all waking time. This permanent visual scanning is principally done with high-speed targeted eye movements called saccades (Yarbus, 1967), that sequentially capture local chunks of the visual scene. This makes the oculo-motor activity an essential element of man and animal behavior, underlying most of daily displacements, movements, instrumental and social interactions.

Scene decoding through action (or "active perception") has attracted strong interest in robotics and artificial vision, for the important redundancy present in the sensory data allows to envisage energy-efficient sensors, scanning little portions only of the total sensory scene. The opportunity to neglect large parts of the sensory scene should mainly be considered when the energy is scarce, as it is the case for drones and robots. It is also relevant in computer vision, where mega-pixel images appeals for selective convolutions, in order to avoid unnecessary matrix products. The example of animal vision thus encourages a more parsimonious approach to robotic and computer vision, *including the control of the sensory flow*. Optimizing the sensor displacements across time may then be a part of robotic control, in combination with goal-oriented operations.

Changing the viewpoint can be seen as a way to leverage ambiguities present in the current visual field. In Aloimonos et al. (1988), the authors show that some ill-posed object recognition problems become well-posed as soon as several views on the same object are considered. A more general perspective is developed in Bajcsy (1988), with a first attempt to interpret active vision in the terms of sequential Bayesian estimation:

> *The problem of Active Sensing can be stated as a problem of controlling strategies applied to the data acquisition process which will depend on the current state of the data interpretation and the goal or the task of the process.*

thus providing a roadmap for the development of active sensory systems.

Work on active vision control is quite scarce until the late 2000's. On the machine learning side, an example of fovea-based visuo-motor control was addressed in Schmidhuber and Huber (1991), with a direct policy learning from gradient descent by using BPTT through a pre-processed forward model. On the biological side, early models from the late nineties consider the case of fovea-based image encoding, ending up in the simplified "pyramidal" focal image encoding model (Kortum and Geisler, 1996). Active vision models were however largely dominated by the *salience* models (Itti and Koch, 2000, 2001; Itti and Baldi, 2005), that were shown consistent with the preferred fixation zones observed in humans. Motor control were however generally bypassed in that case, putting the focus on characterizing the attractiveness of fixation zones rather that explaining the scene decoding process when changing gaze orientation.

In contrast, global scene understanding implies to consider the visual scan-path as a sequential *sampling* of an underlying (covert) sensory scene, given a generative model. Two parallel research tracks adopted and refined this last idea over the last 20 years. On the one side, a *predictive* approach to active vision was originally developed in (Najemnik and Geisler, 2005). It globally complies with the predictive coding framework (Rao and Ballard, 1999) with the current posterior estimate used to anticipate future sensations. Here, appropriate samples should be selected that maximize the expected *decoding accuracy*, that resorts to reduce the number of possible interpretations of the underlying scene, i.e., reduce the expected posterior *entropy* (see Najemnik and Geisler, 2005, 2009; Butko and Movellan, 2010; Friston et al., 2012). It also generalizes to the case of multi-view selection in object search and scene recognition (Potthast et al., 2016). A second research track insists on the formal contribution of action in the *encoding* of the (future) sensory field. This resorts to consider action as a *code* that is later on revealed (decoded) by sensing the effect of action at the visual field (Klyubin et al., 2005; Tishby and Polani, 2011). As such it may be optimized so as to maximize the action read-out capability, allowing to improve both the policy and the data model in the course of learning (Schmidhuber, 2007; Mohamed and Rezende, 2015; Houthooft et al., 2016).

Those different approaches interestingly conduct to develop different action selection *policies* that do not appear mutually compatible in the first place. The decoding accuracy objective encourages actions that provide a consistent belief update, measured at the log likelihood of the data after sampling. This implies to avoid surprising data and prefer actions that bring out a sensory input that is consistent with the initial guess (Friston, 2010). This approach may be referred as the "conservative" approach to action selection. Conversely,the "maximum effect" principle encourages actions that are well discriminated, i.e., that have a visible effect on the sensors. This is formally quantified by the "empowerment" information gain objective (Klyubin et al., 2005; Tishby and Polani, 2011), or by the more informal measures of surprise, like the "Salience" metric (Itti and Baldi, 2005), or the different "curiosity" metrics, like the ones proposed in Schmidhuber (1991), Oudeyer and Kaplan (2008), and Pathak et al. (2017). This second approach may be referred as the "progressive" approach to action selection.

Active vision is thus in need for clarification, in order to develop more effective and principle-grounded action-selection controllers in open environments. This article is an attempt to set the ground for such a unifying framework, making easier both a formal and quantitative comparisons between the different action selection metrics at stake. A fovea-based visuo-motor control setup is used for illustration, that consists in choosing the next saccade in a sequential visual scene decoding task.

A general active scene decoding framework is first developed in section 2.1, under predictive control assumptions, with a generative model explaining how the observed data is shaped by action. Stemming from a partially observed probabilistic framework, the current observation is interpreted as the realization of a *mixed emission density* made of a controlled emitter (i.e., the actuator state) and an uncontrolled one (i.e., the latent state of the environment). Then, when combined with a chain rule-based sequential update, it is shown how the (unobserved) latent state shall be inferred from both the current observation and past inferences memory. Given that "mixed" generative model, a generic active inference framework is developed in section 2.2, with classical action selection metrics recast, showing a clear formal separation between the "Accuracy-based," "Innovation-based," and "Conservation-based" action-selection metric families.

Section 3 gathers both formal and simulation results, providing a comprehensive and consistent interpretation of most existing metrics as maximizing the view-based Information Gain, either in an "optimistic" or in a "pessimistic" fashion. The connection between action-selection metrics and the Information Gain is first formally unfolded in section 3.1. It is shown that a rather straightforward approximation of the Information Gain, known as the Compression Improvement, provides both a general setup to interpret most classic objective functions, and a baseline to provide new effective and principle-grounded objective functions, namely the Information Gain Lower Bound (IGLB) and the Information Gain Upper Bound (IGUB). Then an actual implementation of a sequential fovea-based scene decoding setup is developed in section 3.2, allowing to quantitatively compare those different metrics, and propose new avenues toward parsimonious active vision through computationally-effective model-based prediction.

2. PRINCIPLES AND METHODS

We consider here a *scene decoding task* where an agent has to estimate its environment state, here called the "sensory scene," from sensory samples. The visual scene is organized in objects (or objects parts), whose presence and position is continuously checked by visual inspection. Then, decoding a visual scene through saccades consists in identifying the ensemble through the sequential foveation of parts of the scene only.

2.1. A Mixed Generative Model

The active inference approach relies on a longstanding history of probabilistic modeling in signal processing and control (Kalman, 1960; Baum and Petrie, 1966). The physical world takes the form of a random process that is the cause of the sensory stream. This process is not visible in itself but only sensed through (non reliable) sensors, providing a sequence of observations over time. The inference problem consists in identifying the cause of the observations (i.e., the state of the environment), given the generative model. The result of the inference is itself a probability density over the hidden states (the posterior probability), that is obtained through inverting the model (from the observations toward the hidden states).

2.1.1. One Scene, Many Views

A feedback control framework is composed of an actor and an environment. The actor and the environment interact according to a feedback loop. The actor can act on the environment through its effectors, and sense the state of the environment through its sensors. The state of the environment as well as the state of the agent can change over time. The state of the environment is described by a state vector $\boldsymbol{s} \in \mathcal{S}$. The signal $\boldsymbol{x}$ that is measured on the sensors is called the *sensory field*. It is interpreted as a measure made by the sensors, that is causally related to the current state $\boldsymbol{s}$.

We consider here an organization of the environment in objects (or object parts), whose presence and position is continuously checked by sensori-motor inspection. In a (discrete) Markovian framework, the state $\boldsymbol{s}$ in which the physical system is found at present depend both on its previous state (say $\boldsymbol{s}_0$) and on a preceding motor command $\boldsymbol{a}$. The transition from $\boldsymbol{s}_0$ to $\boldsymbol{s}$ is reflected in a *transition* probability that embodies the deterministic and non-deterministic effects of the command $\boldsymbol{a}$ in the form of a conditional probability:

$$\boldsymbol{s} \sim \Pr(S|\boldsymbol{a}, \boldsymbol{s}_0) \tag{1}$$

The signal $\boldsymbol{x}$ measured on the sensors is interpreted as an effect of the current state $\boldsymbol{s}$. Once again the deterministic and non-deterministic effects are reflected in a conditional probability:

$$\boldsymbol{x} \sim \Pr(X|\boldsymbol{s}) \tag{2}$$

that is said the *sensory emission* process. The combination of (1) and (2) is the *generative process* that is the cause of the sensory field. Consider now the cause $\boldsymbol{s}$ of the current visual field $\boldsymbol{x}$ is both the object identity $\boldsymbol{o}$, its position in the peripheral space $\boldsymbol{y}$, and the current body orientation $\boldsymbol{u}$, i.e., $\boldsymbol{s} = (\boldsymbol{y}, \boldsymbol{o}, \boldsymbol{u})$, with $\boldsymbol{x} \sim \Pr(X|\boldsymbol{y}, \boldsymbol{o}, \boldsymbol{u})$ the sensory emission. Here each variable accounts for a distinct degree of freedom responsible for the sensory emission.

Then we propose to split the generative process in two parts, namely the controlled generative process and the uncontrolled generative process. This separation is consistent with the "hidden state"/"hidden control" distinction stated in Friston et al. (2012). The controlled emitter is $\boldsymbol{u}$ while the uncontrolled emitter is $(\boldsymbol{y}, \boldsymbol{o})$. Moreover, for greater simplicity, $(\boldsymbol{y}, \boldsymbol{o})$ is here reduced to a single variable $\boldsymbol{z} = (\boldsymbol{y}, \boldsymbol{o})$, so that the generic uncontrolled state $\boldsymbol{z}$ may report for every possible composition of object identity in space (or more generally every composition of a pose and an identity). The controlled emitter $\boldsymbol{u}$ refers to the state of a motor apparatus, e.g., to the spatial distribution of the different mobile segments of an articulated body. The uncontrolled latent emitter $\boldsymbol{z}$ refers to the remaining part of the physical world, i.e., the "environment."

This restricted setup, that separates a body and an environment in the form of two independent processes, provides a substantial simplification to the estimation problem at stake (see **Appendix A** in Supplementary Material). The controlled transition is assumed to be relatively "fast" in comparison with the uncontrolled one (for e.g., saccades can be realized in a 100–200 ms interval). Consistently with the "end-effector" ballistic control setup (Mussa-Ivaldi and Solla, 2004), the motor command $\boldsymbol{a}$ is thus assimilated with a setpoint (or posture) $\boldsymbol{u}$ in the actuator space. Under that perspective, the motor command acts on the sensors position and orientation so as to achieve a certain perspective (or view) over the external scene, here called a *viewpoint*.

Finally, both $\boldsymbol{x}$ (the view) and $\boldsymbol{z}$ (the *latent state*) are the realization of a generative model parametrized by $\boldsymbol{u}$ (the viewpoint), i.e.,

$$\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u} \sim \Pr(X|Z, \boldsymbol{u}), \Pr(Z) \tag{3}$$

with $\Pr(Z)$ the *prior*, and each different motor command $\boldsymbol{u}$ providing a different sample over the same underlying distribution. With that respect, the action $\boldsymbol{u}$ is also interpreted as a *sampling* operation.

2.1.2. Sequential Bayesian Inference

With a generative model comes the possibility to *infer* the latent state of the physical system from the observation using Bayes rule:

$$\Pr(Z|\boldsymbol{x}, \boldsymbol{u}) = \frac{\Pr(\boldsymbol{x}|Z, \boldsymbol{u})\Pr(Z)}{\Pr(\boldsymbol{x}|\boldsymbol{u})} \tag{4}$$

with $\Pr(Z|\boldsymbol{x}, \boldsymbol{u})$ the *posterior* probability (over the latent states), whose order 2 moment informs on the estimation accuracy : the narrower the distribution, the more accurate the latent state prediction.

In a visual scene decoding task, a single latent state $\boldsymbol{z}$ is observed through a series of viewpoints $\boldsymbol{u}, \boldsymbol{u}', \boldsymbol{u}''$, ... This sequence of observations should ultimately provide a final estimate $\hat{q}(Z)$, with a single cause $\hat{\boldsymbol{z}}$ dominating the other ones, allowing to reach a final decision. The chaining of the posterior to the role of the prior in the next inference step is a classical property of sequential Bayesian inference. When generalized to many observations: $(\boldsymbol{x}|\boldsymbol{u}), (\boldsymbol{x}'|\boldsymbol{u}'), ..., (\boldsymbol{x}^{(n)}|\boldsymbol{u}^{(n)})$, the final posterior $q^{(n)}(Z)$ writes:

$$q^{(n)}(Z) \propto \Pr(\boldsymbol{x}|Z, \boldsymbol{u}) \times \Pr(\boldsymbol{x}'|Z, \boldsymbol{u}') \times ... \times \Pr(\boldsymbol{x}^{(n)}|Z, \boldsymbol{u}^{(n)}) \times \Pr(Z) \tag{5}$$

which allows to approach the latent state $\boldsymbol{z}$ from many samples of (3), each sample providing more evidence. When the sampling is done incrementally (Wald, 1945), the $\boldsymbol{u}$'s and $\boldsymbol{x}$'s do not need to be stored in the process. At step n, only $q^{(n-1)}$ (the current "belief") needs to be memorized to estimate $q^{(n)}$, i.e.,

$$q^{(n)}(Z) \propto \Pr(\boldsymbol{x}^{(n)}|Z, \boldsymbol{u}^{(n)}) \times q^{(n-1)}(Z) \tag{6}$$

2.2. Active Vision and Predictive Control

Consider an agent having to estimate its environment state $\boldsymbol{z}$ from sampling it from different viewpoints. We here suppose that a generative model p is given to the agent. Depending on the current viewpoint $\boldsymbol{u}$, a different view $\boldsymbol{x}$ is observed at the sensors. So, each different command $\boldsymbol{u}$ provokes a different observation, and thus a different estimation of the latent state. It is thus worth to question what is the optimal choice for $\boldsymbol{u}$ in order to maximize the accuracy of the posterior estimate? That turns to minimize the number of samples so as to provide an accurate estimate. This approach to inference is called *active sampling* in Friston et al. (2012), for the choice of $\boldsymbol{u}$ determines the sensory sample $\boldsymbol{x}$ that is observed, conditioning the final posterior estimate. It was originally developed by Najemnik and Geisler (2005) to the case of human visual search modeling (finding a target feature in an image, i.e., the "find Waldo" task).

A baseline sampling strategy is to choose $\boldsymbol{u}$ at random and condition the posterior estimate on this random action. More elaborate strategies consider the past observations to choose the most promising action $\hat{\boldsymbol{u}}$. The knowledge about past observations being here absorbed in single posterior distribution $q^{(n-1)}$, the problem turns out to design a *controller C* which, given a context $q^{(n-1)}$, sets up an action $\hat{\boldsymbol{u}} = C(q^{(n-1)})$. Here the role of the controller is not to achieve a goal-oriented task, but to render the estimation of the latent state more accurate. The controller is said *perception-driven*.

The design of such a controller is not straightforward. On contrary to classical control, there is not definite setpoint $\boldsymbol{z}^*$ to which the controller is supposed to drive the external process (through model inversion for instance). By design, the actual latent state $\boldsymbol{z}$ is not visible as such and can not be compared to the inferred posterior. In order to estimate how good a motor command is, one needs to provide an estimate of the value-of-action (regarding scene understanding). There is currently no consensus about what a good value is regarding the scene decoding task.

A general strategy is thus to establish an *action selection metric*, taking either the form of an *objective function f* or a loss ℓ, that conveys a quantitative estimation of the action's contribution to the inference accuracy (resp. imprecision). Once the objective function established, a simple control strategy is to choose the action that maximizes the objective (resp. minimizes the loss), i.e.,:

$$\hat{\boldsymbol{u}} = \underset{\boldsymbol{u}\in\mathcal{U}}{\text{argmin}}\ \ell(\boldsymbol{u}) \Big/ \underset{\boldsymbol{u}\in\mathcal{U}}{\text{argmax}}\, f(\boldsymbol{u}) \tag{7}$$

Many such objective functions are proposed in the literature. They are generally referred as an *intrinsic* motivation (Oudeyer and Kaplan, 2008) by contrast with the *extrinsic* motivation that relates to the classical rewards in reinforcement learning (Sutton and Barto, 1998). Several such intrinsic reward candidates have been developed in recent years in the scene decoding context. Some of them are presented in the next paragraphs. The original formulas have been recast to show their formal correspondences, but also highlight some manifest differences between them.

2.2.1. Accuracy-Based Action Selection

Given a generative model $p(X, U, Z)$, like the one described in section 2.1, the predictive approach to perception-driven control (Najemnik and Geisler, 2005) relies on predicting an *accuracy* measure $A(\boldsymbol{x}, \boldsymbol{u}; q^{(n-1)})$ to choose action. The accuracy tells how good the model is at predicting $\boldsymbol{z}$ (here the target position) when viewing $\boldsymbol{x}$ at position $\boldsymbol{u}$, knowing $q^{(n-1)}$ (the estimated posterior at step $n-1$).

If the agent has to choose an action $\boldsymbol{u} \in \mathcal{U}$, knowing only $q^{(n-1)}$, the *predicted* accuracy attached to $\boldsymbol{u}$ is:

$$\begin{aligned}\bar{A}(\boldsymbol{u}; q^{(n-1)}) &= \mathbb{E}_{\boldsymbol{z}\sim q^{(n-1)}(Z), \boldsymbol{x}\sim p(X|\boldsymbol{z},\boldsymbol{u})}\left[A(\boldsymbol{x}, \boldsymbol{u}; q^{(n-1)})\right] \\ &= \sum_{z\in\mathcal{Z}} q^{(n-1)}(\boldsymbol{z}) \int_{\mathcal{X}} A(\boldsymbol{x}, \boldsymbol{u}; q^{(n-1)}) p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{u}) d\boldsymbol{x}\end{aligned}$$

and the optimal action to choose is:

$$\hat{\boldsymbol{u}} = \underset{\boldsymbol{u}\in\mathcal{U}}{\text{argmax}}\ \bar{A}(\boldsymbol{u}; q^{(n-1)}) \tag{8}$$

In order to render the computation tractable, a sample is generally used to estimate the predicted accuracy, i.e., $\mathbb{E}_p[f(\boldsymbol{x})] \simeq f(\tilde{\boldsymbol{x}})$, with $\tilde{\boldsymbol{x}} \sim p(\boldsymbol{x})$.

The accuracy metric used in the original paper was an *ad-hoc* one (Najemnik and Geisler, 2005), but turned out to be consistent with minimizing the *posterior entropy* (Najemnik and Geisler, 2009), i.e.,:

$$A(\boldsymbol{x}, \boldsymbol{u}; q^{(n-1)}) = -H(q^{(n)}) = \sum_{z \in \mathcal{Z}} q^{(n)}(\boldsymbol{z}) \log q^{(n)}(\boldsymbol{z})$$

with: $q^{(n)}(\boldsymbol{z}) \propto p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{u}) \times q^{(n-1)}(\boldsymbol{z})$, so that:

$$\hat{\boldsymbol{u}} = \underset{\boldsymbol{u} \in \mathcal{U}}{\text{argmin}} \, \mathbb{E}_{\boldsymbol{z} \sim q^{(n-1)}(Z), \boldsymbol{x} \sim p(X|\boldsymbol{z}, \boldsymbol{u})} \left[H(q^{(n)}) \right]$$

which makes sense for a low entropy of the posterior is expected when the estimated posterior accuracy is high.

This approach to optimal visual sampling was further on linked to an "Infomax" principle(posterior Mutual Information maximization) in Butko and Movellan (2010), taking:

$$A_{\text{INFOMAX}}(\boldsymbol{x}, \boldsymbol{u}; q^{(n-1)}) \equiv I(Z; \boldsymbol{x}|\boldsymbol{u}; q^{(n-1)}) = H(q^{(n-1)}) - H(q^{(n)}) \quad (9)$$

with $H(q^{(n-1)}) \equiv H(Z|q^{(n-1)})$ and $H(q^{(n)}) \equiv H(Z|\boldsymbol{x}, \boldsymbol{u}; q^{(n-1)})$, which also turns out to minimize $H(q^{(n)})$ solely for $H(q^{(n-1)})$ is independent of $\boldsymbol{u}$. The Infomax (or posterior entropy minimization) approach generally makes sense for it implicitly relies on the chaining from $q^{(n-1)}$ to $q^{(n)}$, that considers that if $p(\boldsymbol{x}|Z, \boldsymbol{u})$ is consistent with $q^{(n-1)}(Z)$, then the issued posterior entropy should be lower than if $p(\boldsymbol{x}|Z, \boldsymbol{u})$ is at odd with $q^{(n-1)}(Z)$. The model is expected to choose the action that may confirm the initial assumption, though there is no formal comparison between $q^{(n-1)}$ and $q^{(n)}$. It is thus potentially vulnerable to model outliers with $q^{(n)}$ having both a low entropy and being inconsistent with $q^{(n-1)}$.

2.2.2. Innovation-Based Action Selection

Another quantity of interest is the so-called *Bayesian surprise* or *Salience* (Itti and Baldi, 2005) defined as the Kullback-Leibler divergence between an actual view $\boldsymbol{x}$ and a model $\boldsymbol{z}$. In the original "bottom-up" setup, only local statistics are formed over small image patches of a given image, with $\boldsymbol{u}$ the index of a patch and $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{u})$ the features inferred from the data actually observed at $\boldsymbol{u}$. For each patch $\boldsymbol{u}$, the Salience of the actual view $\boldsymbol{x}$ given the model is:

$$S(\boldsymbol{x}, \boldsymbol{u}) = \text{KL}(p(Z|\boldsymbol{x}, \boldsymbol{u})||p(Z))$$

with $\text{KL}(p_1||p_2) = \sum_z p_1(\boldsymbol{z}) \log \frac{p_1(\boldsymbol{z})}{p_2(\boldsymbol{z})}$ the Kullback-Leibler divergence between p_1 and p_2, interpreted here as a measure of the *in*consistency between a (viewpoint independent) model p and the data. A high salience reflects a strong inconsistency with the model, while a low salience reflects a strong consistency with the model. According to Itti and Baldi, the regions that have a high Bayesian surprise are the ones that attract the sight the most. The calculation of $S(\boldsymbol{x}, \boldsymbol{u})$ at each location $\boldsymbol{u}$ forms a *saliency map* that is then considered as a prediction of where the sight will most likely be attracted (high values most probably attract the sight, low values less probably do). The saliency model has a strong explanatory power and provides among the best fit with the actual preferred fixation zones observed in humans. Its scalability moreover provides straightforward applications in image and video compression (Wang et al., 2003; Guo and Zhang, 2010).

Generalized to the sequential setup, the saliency metric becomes:

$$S(\boldsymbol{x}, \boldsymbol{u}; q^{(n-1)}) = \text{KL}(q^{(n)}||q^{(n-1)}) \quad (10)$$

with $q^{(n-1)}$ considered as the data model and $q^{(n)}$ the posterior estimated at $(\boldsymbol{x}, \boldsymbol{u})$, knowing $q^{(n-1)}$. Through maximizing the KL divergence between the previous and the current scene interpretation, the Saliency objective is found here to promote the most *conflicting* observation regarding previous assumptions, entailing finding innovative interpretations of the current scene.

Put in a predictive form, it gives:

$$\hat{\boldsymbol{u}} = \underset{\boldsymbol{u} \in \mathcal{U}}{\text{argmax}} \, \mathbb{E}_{\boldsymbol{z} \sim q^{(n-1)}, \boldsymbol{x} \sim p(X|\boldsymbol{z}, \boldsymbol{u})} \left[\text{KL}(q^{(n)}||q^{(n-1)}) \right]$$

with the predictive Saliency promoting alternate future interpretations regarding the current interpretation. This entails searching for model inconsistencies or model contradicting predictions, making the Saliency a model consistency check metric.

2.2.3. Conservation-Based Action Selection

At last, the Variational Free Energy based (VFE) active inference setup (Friston, 2010; Friston et al., 2012) considers the general tendency of the brain to counteract surprising and unpredictable sensory events through minimizing the VFE with action (see **Appendix B** in Supplementary Material). In our sequential setup, it writes:

$$F(\boldsymbol{x}|\boldsymbol{u}) = -\log p(\boldsymbol{x}|\boldsymbol{u}) + \text{KL}(q(Z)||p(Z|\boldsymbol{x}, \boldsymbol{u})) \quad (11)$$

From the predictive perspective, stemming from $q^{(n-1)}$ as the current scene interpretation, Friston et al. (2017) propose a *predictive VFE* that writes in our sequential setup like :

$$\bar{F}(\boldsymbol{u}; q^{(n-1)}) = \mathbb{E}_{\boldsymbol{z} \sim q^{(n-1)}(Z), \boldsymbol{x} \sim p(X|\boldsymbol{z}, \boldsymbol{u})} \left[-\log p(\boldsymbol{x}|\boldsymbol{u}; q^{(n-1)}) + \text{KL}(q^{(n-1)}||q^{(n)}) \right] \quad (12)$$

with $q^{(n)}$ the predicted posterior and $\text{KL}(q^{(n-1)}||q^{(n)})$ quantifying the scene interpretation update made by interpreting the scene with $q^{(n)}$ instead of $q^{(n-1)}$. It is said the "epistemic cost"[1]. In that setup, minimizing the Free-Energy is consistent with minimizing $\text{KL}(q^{(n-1)}||q^{(n)})$ estimated as:

$$\text{KL}(q^{(n-1)}||q^{(n)}) = \mathbb{E}_{\boldsymbol{z} \sim q^{(n-1)}(Z)} \left[\log q^{(n-1)}(\boldsymbol{z}) - \log q^{(n)}(\boldsymbol{z}) \right]$$

Put in a predictive form, the selection of action finally relies on reducing the predicted log ratio, i.e., :

$$\hat{\boldsymbol{u}} = \underset{\boldsymbol{u} \in \mathcal{U}}{\text{argmin}} \, \mathbb{E}_{\boldsymbol{z} \sim q^{(n-1)}(Z), \boldsymbol{x} \sim p(X|\boldsymbol{z}, \boldsymbol{u})} \left[\log q^{(n-1)}(\boldsymbol{z}) - \log q^{(n)}(\boldsymbol{z}) \right] \quad (13)$$

[1] or negative epistemic value.

which may minimize the *epistemic cost*[2].

On contrary to the Infomax objective (section 2.2.1), minimizing the epistemic cost selects the predicted posterior having the highest consistency with the current posterior, which may prevent from model inconsistencies that may incidently "hack" the posterior entropy (see section 2.2.1). Minimizing $\text{KL}(q^{(n-1)}||q^{(n)})$ thus corresponds to a *conservative* approach to the scene interpretation that is *minimally* vulnerable to outliers, i.e., that minimizes the risk of a conflicting interpretation.

The epistemic cost is moreover at odd with the Saliency objective (section 2.2.2) seeking the maximal *in*consistency between the cumulated posterior and the current posterior. It is obvious here that the Free Energy minimization and the Saliency maximization are antithetic objectives, and no consensus is currently observed in the literature about which objective should prevail (though Infomax generally preferred in scene decoding and saliency/surprise preferred in sparse reinforcement learning).

3. RESULTS

3.1. View-Based Information Gain Metrics

The structure of the problem (many views on the same scene) implies that different observations should share a common information corresponding to the actual (covered) sensory scene. We take here benefit of the viewpoint-based variational encoding setup (seen in section 2.2.3 and **Appendix B** in Supplementary Material) to propose a new quantification the *mutual information* shared across different sensory fields, locally estimated with a view-based Information Gain metric. It is shown here that a rather straightforward approximation of the information gain, known as the Compression Improvement, provide both a general setup to interpret most classic objective functions, and a baseline to provide new effective and principle-grounded objective functions.

3.1.1. Definitions

3.1.1.1. View-based mutual information and information gain

The sharing of information between two sensory fields $\boldsymbol{x}|\boldsymbol{u}$ and $\boldsymbol{x}'|\boldsymbol{u}'$ should be quantified by their *Mutual Information*. The general idea is that two samples may provide more insight about the hidden sensory scene than a single one (three samples should provide even more, etc.). Consider $\boldsymbol{x}|\boldsymbol{u}$ as the initial sensory sample and $\boldsymbol{x}'|\boldsymbol{u}'$ as an additional sample providing new evidence about how interpret the initial view $\boldsymbol{x}$. The view-based mutual information writes:

$$I((X|\boldsymbol{u});(X'|\boldsymbol{u}')) = H(X|\boldsymbol{u}) - H(X|\boldsymbol{u},X',\boldsymbol{u}') \quad (14)$$
$$\simeq \mathbb{E}_{X,X'}\left[-\log p(X|\boldsymbol{u}) + \log p(X|\boldsymbol{u},X',\boldsymbol{u}')\right] \quad (15)$$

with :

(*i*) $p(\boldsymbol{x}|\boldsymbol{u}',\boldsymbol{x}',\boldsymbol{u}) \triangleq \sum_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{u})p(\boldsymbol{z}|\boldsymbol{x}',\boldsymbol{u}')$ the *post-sample* likelihood, i.e., the retrospective likelihood of having seen $\boldsymbol{x}$ at $\boldsymbol{u}$ knowing now that $\boldsymbol{x}'$ is observed at $\boldsymbol{u}'$, and

(*ii*) $-\log p(\boldsymbol{x}|\boldsymbol{u}) + \log p(\boldsymbol{x}|\boldsymbol{u}',\boldsymbol{x}',\boldsymbol{u})$ the post-sample *Information Gain* (see also Tishby and Polani (2011)), that is a local estimator of the views mutual information at $X = \boldsymbol{x}$ and $X' = \boldsymbol{x}'$, given the model p:

$$\text{IG}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}') = -\log p(\boldsymbol{x}|\boldsymbol{u}) + \log p(\boldsymbol{x}|\boldsymbol{u}',\boldsymbol{x}',\boldsymbol{u}) \quad (16)$$

3.1.1.2. Conditional reconstruction cost

Stemming from the sequential Bayes posterior update formula:

$$p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}') = \frac{p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{u})p(\boldsymbol{z}|\boldsymbol{x}',\boldsymbol{u}')}{p(\boldsymbol{x}|\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}')} \quad (17)$$

It can be shown that the negative log likelihood of $\boldsymbol{x}$ after seeing both $\boldsymbol{x}$ and $\boldsymbol{x}'$ is bounded from above by:

$$-\log p(\boldsymbol{x}|\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}') \leq -\sum_{z} q'(z)\log p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{u})\frac{p(\boldsymbol{z}|\boldsymbol{x}',\boldsymbol{u}')}{q'(z)}$$
$$= \mathbb{E}_{\boldsymbol{z}\sim q'}\left[-\log p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{u})\right] + \text{KL}(q'(Z)||p(Z|\boldsymbol{x}',\boldsymbol{u}')) \quad (18)$$
$$= -\log p(\boldsymbol{x}|\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}') + \text{KL}(q'(Z)||p(Z|\boldsymbol{x},\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}')) \quad (19)$$
$$\triangleq F(\boldsymbol{x}|\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}')$$

which establishes $F(\boldsymbol{x}|\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}')$ as the post-sample *conditional reconstruction cost* (or *conditional Free Energy* – the two are synonyms), with $q'(\boldsymbol{z})$ expectedly approaching $p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}')$ after optimization. From a variational perspective, the passing from $q(Z) \simeq p(Z|\boldsymbol{x},\boldsymbol{u})$ to $q'(Z) \simeq p(Z|\boldsymbol{x},\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}')$ is the *variational posterior update*, and the passing from $F(\boldsymbol{x}|\boldsymbol{u})$ toward $F(\boldsymbol{x}|\boldsymbol{u}',\boldsymbol{x}',\boldsymbol{u})$ is the *reconstruction cost update*.

3.1.1.3. Compression improvement

An approximation of the Information Gain (IG), known as the *Compression Improvement* (CI) was proposed in Schmidhuber (2007) and Houthooft et al. (2016). In our view-based setup, it writes :

$$\text{CI}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}') = F(\boldsymbol{x}|\boldsymbol{u}) - F(\boldsymbol{x}|\boldsymbol{u},\boldsymbol{x}',\boldsymbol{u}') \quad (20)$$

There comes the possibility to optimize the *next* sampling $\boldsymbol{u}'$ through maximizing the CI as a proxy for the IG. It happens to be equivalent with minimizing the post-sample Free Energy, consistently with Friston et al. (2012)'s intuition.

3.1.2. The Sequential Information Gain and Its Approximations

Extending now to the sequential setup, the contribution of $\boldsymbol{u}^{(n)}$ in understanding the scene is measured by a change in the reconstruction cost F *before* and *after* reading $\boldsymbol{x}^{(n)}|\boldsymbol{u}^{(n)}$.

- Before reading $\boldsymbol{x}^{(n)}$, the reconstruction cost at $\boldsymbol{x}^{(n-1)}$ writes:

$$F(\boldsymbol{x}^{(n-1)}|\boldsymbol{u}^{(n-1)};q^{(n-2)}) = \mathbb{E}_{\boldsymbol{z}\sim q}\left[-\log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{z},\boldsymbol{u}^{(n-1)})\right] + \text{KL}(q||q^{(n-2)}) \quad (21)$$
$$= -\log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{u}^{(n-1)};q^{(n-2)}) + \text{KL}(q||q^{(n-1)}) \quad (22)$$

[2]It is to be noticed that the Kullback-Leibler divergence is here absorbed in the general expectation over the $\boldsymbol{x}$'s and the $\boldsymbol{z}$'s.

- After reading $\boldsymbol{x}^{(n)}$, it writes:

$$F(\boldsymbol{x}^{(n-1)}|\boldsymbol{u}^{(n-1)}, \boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)}; q^{(n-2)}) = \mathbb{E}_{\boldsymbol{z}\sim q'}\left[-\log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{z}, \boldsymbol{u}^{(n-1)})\right] + \mathrm{KL}(q'||q^{(n;n-2)}) \tag{23}$$

$$= -\log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{u}^{(n-1)}; q^{(n;n-2)}) + \mathrm{KL}(q'||q^{(n)}) \tag{24}$$

with :

$$q^{(n;n-2)}(Z) \propto p(Z|\boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)})q^{(n-2)}(Z) \tag{25}$$

Before reading $\boldsymbol{x}^{(n)}$, the optimal reconstruction cost is attained at $q = q^{(n-1)}$. After reading $\boldsymbol{x}^{(n)}$, the optimal reconstruction cost is attained at $q' = q^{(n)}$. From subtracting (23) from (21), the CI writes :

$$\mathrm{CI}^{(n)} = \mathbb{E}_{\boldsymbol{z}\sim q}\left[-\log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{z}, \boldsymbol{u}^{(n-1)})\right] + \mathrm{KL}(q||q^{(n-2)}) + \mathbb{E}_{\boldsymbol{z}\sim q'}\left[\log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{z}, \boldsymbol{u}^{(n-1)})\right] - \mathrm{KL}(q'||q^{(n;n-2)}) \tag{26}$$

Knowing that q and q' are free parameters, taking $q = q'$ provides a strong simplification of the above formula, further on referred as the *approximate CI*:

$$\tilde{\mathrm{CI}}^{(n)} = \mathrm{KL}(q||q^{(n-2)}) - \mathrm{KL}(q||q^{(n;n-2)}) \tag{27}$$

$$= \mathbb{E}_{z\sim q}\left[\log p(\boldsymbol{z}|\boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)})\right] + c \tag{28}$$

with c a constant. The information gain is here approached with the opposite of the cross-entropy cost $\tilde{\mathrm{CI}}^{(n)} = -H(q(Z), p(Z|\boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)})) + c$.

3.1.2.1. Information gain lower bound (IGLB)

Maximizing the CI however provides no formal guarantee the IG will be maximized. The reconstruction cost is indeed an upper bound of the negative log evidence, but the difference of two reconstruction costs is neither an upper bound or a lower bound of the IG (so that it may either underestimate or overestimate the IG). It happens that, contrarily to the original CI formula (Equation 26), the approximate CI (Equation 28) provides ways to establish firm bounds regarding the Information Gain estimate. Taking for instance $q = q^{(n-1)}$ (the pre-sample posterior), and subtracting (24) from (22), the approximate CI writes:

$$\tilde{\mathrm{CI}}^{(n)}_{q=q^{(n-1)}} = -\log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{u}^{(n-1)}; q^{(n-2)}) + \log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{u}^{(n-1)}; q^{(n;n-2)}) - \mathrm{KL}(q^{(n-1)}||q^{(n)}) \tag{29}$$

making the approximate CI objective a *lower bound* of the IG (Equation 16), for it *underestimates* the IG by an amount equal to $\mathrm{KL}(q^{(n-1)}||q^{(n)})$, thus providing a principled objective to optimize action, with a simple objective function defined here as the *Information Gain Lower Bound* (IGLB):

$$\tilde{\mathrm{CI}}^{(n)}_{q=q^{(n-1)}} \triangleq \mathrm{IGLB}(\boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)}, q^{(n-1)}) = \mathbb{E}_{z\sim q^{(n-1)}}\left[\log p(\boldsymbol{z}|\boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)})\right] + c \tag{30}$$

It is maximal when $p(Z|\boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)}) \simeq q^{(n-1)}(Z)$, i.e., when the current posterior $p(Z|\boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)})$ and the past cumulated posterior $q^{(n-1)}$ are highly consistent.

If now $\boldsymbol{u}$ is at choice *before* sampling $\boldsymbol{x}$, it is sensible to maximize the predicted IGLB to maximize the code consistency, i.e.,:

$$\hat{\boldsymbol{u}} = \underset{\boldsymbol{u}\in\mathcal{U}}{\mathrm{argmax}}\ \mathbb{E}_{z\sim q^{(n-1)}; \boldsymbol{x}\sim p(X|\boldsymbol{z},\boldsymbol{u}))}\left[\log p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{u})\right] \tag{31}$$

The amount of the underestimation, i.e., $\mathrm{KL}(q^{(n-1)}||q^{(n)})$ defines the epistemic cost (see section 2.2.3) as the *IGLB bias*, that should be minimized through action (see Equation 12). Minimizing this term means reducing the underestimation made about the future information gain *before posterior update*. This means searching for "accurate" Information Gains approximations, rather than maximizing the Information Gain itself, that links the IGLB with "safe" or "conservative" action selection policies.

At last, the IGLB objective containing both the IG objective and the (negative) epistemic cost, maximizing the IGLB is expected to both promote high information gains and prevent for conflicting predictions, making it a "conservative" information-seeking objective.

3.1.2.2. Information gain upper bound (IGUB)

For completeness, instantiating q with $q^{(n)}$ gives a different objective that writes:

$$\tilde{\mathrm{CI}}^{(n)}_{q=q^{(n)}} = -\log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{u}^{(n-1)}; q^{(n-2)}) + \log p(\boldsymbol{x}^{(n-1)}|\boldsymbol{u}^{(n-1)}; q^{(n;n-2)}) + \mathrm{KL}(q^{(n)}||q^{(n-1)}) \tag{32}$$

making it an upper bound of the IG that *overestimates* the information gain by an amount equal to $\mathrm{KL}(q^{(n)}||q^{(n-1)})$.

If now $\boldsymbol{u}$ is at choice *before* sampling $\boldsymbol{x}$, maximizing the approximate CI gives:

$$\hat{\boldsymbol{u}} = \underset{\boldsymbol{u}\in\mathcal{U}}{\mathrm{argmax}}\ \mathbb{E}_{z\sim q^{(n-1)}; \boldsymbol{x}\sim p(X|\boldsymbol{z},\boldsymbol{u}))}\left[\mathbb{E}_{z'\sim q^{(n)}} \log p(\boldsymbol{z}'|\boldsymbol{x}, \boldsymbol{u})\right]$$

$$= \underset{\boldsymbol{u}\in\mathcal{U}}{\mathrm{argmax}}\ \mathbb{E}_{z\sim q^{(n-1)}; \boldsymbol{x}\sim p(X|\boldsymbol{z},\boldsymbol{u}))}\left[-H(q^{(n)}(Z), p(Z|\boldsymbol{x}, \boldsymbol{u}))\right] \tag{33}$$

$$= \underset{\boldsymbol{u}\in\mathcal{U}}{\mathrm{argmax}}\ \mathbb{E}_{z\sim q^{(n-1)}; \boldsymbol{x}\sim p(X|\boldsymbol{z},\boldsymbol{u}))}\left[-H(q^{(n)}) - \mathrm{KL}(q^{(n)}(Z)||p(Z|\boldsymbol{x}, \boldsymbol{u}))\right] \tag{34}$$

that interestingly combines the Infomax objective (Equation 9) with a consistency objective. This mixed objective is further on referred as the *Information Gain Upper Bound* (IGUB):

$$\mathrm{IGUB}(\boldsymbol{x}, \boldsymbol{u}, q^{(n-1)}) \triangleq -H(q^{(n)}) - \mathrm{KL}(q^{(n)}(Z)||p(Z|\boldsymbol{x}, \boldsymbol{u})) + c \tag{35}$$

The IGUB bias, i.e., $\mathrm{KL}(q^{(n)}||q^{(n-1)})$, interestingly fits the Saliency objective (Equation 10), allowing to interpret the Saliency as the amount of the *overestimation* made *after posterior update* by considering (28) instead of (26).

Maximizing the Saliency is here interpreted as promoting the most optimistically biased Information Gain approximation, irrespectively of the Information Gain itself, making it a "risk-seeking" (rather than information seeking) objective, that links with exploratory behavior. At last, the IGUB objective containing both the IG objective and the Saliency, maximizing the IGUB is expected to both promote high information gains and promote conflicting predictions, making it an "optimistic" information-seeking objective.

To conclude, remarkable here is the fact that both the Saliency objective, the Infomax objective and the Free Energy epistemic cost show a consistent inclusion in a more general approximate Information Gain maximization principle. The final set of metrics to be compared in next section are finally displayed in **Table 1**.

3.2. Fovea-Based Visual Scene Decoding

We now turn to an actual implementation of a saccadic visual scene decoding setup. The image (i.e., the visual scene to decode) is analyzed through a finite sequence of local *foveated* visual samples. The control problem consists in choosing the next saccade, given the past observations and the current scene interpretation. The final decoding means identifying to which category the image belongs to (here the label of the MNIST digits dataset). In that setup,

- The viewpoint $\boldsymbol{u}$ is defined as the current gaze orientation on the image (i.e., the central fixation setpoint in pixel coordinates),
- The view $\boldsymbol{x}|\boldsymbol{u}$ is a retinocentric visual sample measured at position $\boldsymbol{u}$, with central magnification and peripheral blurring,
- The latent state $\boldsymbol{z}$ (visual scene interpretation) is the category of the image (here a digit label), to be guessed from several visual samples.

3.2.1. Fovea-Based Vision

3.2.1.1. Pyramidal fovea-based visual observation

In superior vertebrates, two principal tricks are used to minimize sensory resource consumption in scene exploration. The first trick is the foveated retina, that concentrates the photoreceptors at the center of the retina, with a more scarce distribution at the periphery (Osterberg, 1935). A foveated retina allows both treating central high spatial frequencies, and peripheral low spatial frequencies at a single glance (i.e., process several scales in parallel). The second trick is the sequential saccadic scene exploration, already mentioned, that allows to grab high spatial frequency information where it is necessary (serial processing).

The baseline vision model we propose relies first on learning local foveated views on images. Consistently with (Kortum and Geisler, 1996; Wang et al., 2003), we restrain here the foveal transformation to its core algorithmic elements, i.e., the local compression of an image according to a particular focus. Our foveal image compression thus rests on a "pyramid" of 2D Haar wavelet coefficients placed at the center of sight. Taking the example of the MNIST dataset[3] (see **Figure 1A**), we first transform the original images according to a 5-levels wavelet decomposition (see **Figure 1B**). We then define a viewpoint $\boldsymbol{u} = (i, j, h)$ as a set of 3 coordinates, with i the row index, j the column index and h the spatial scale. Each $\boldsymbol{u}$ generates a visual field made of three wavelet coefficients $\boldsymbol{x}_{i,j,h} \triangleq \boldsymbol{x}|(i, j, h) \in \mathbb{R}^3$, obtained from an horizontal, a vertical and an oblique filter at location (i, j) and scale h. The multiscale visual information $\boldsymbol{x}_{i,j} \triangleq \boldsymbol{x}|(i, j) \in \mathbb{R}^{15}$ available at coordinates (i, j) corresponds to a set of 5 coefficient triplets, namely:

$$\boldsymbol{x}_{i,j} = \{\boldsymbol{x}_{i,j,5}, \boldsymbol{x}_{\lfloor i/2\rfloor,\lfloor j/2\rfloor,4}, \boldsymbol{x}_{\lfloor i/4\rfloor,\lfloor j/4\rfloor,3}, \boldsymbol{x}_{\lfloor i/8\rfloor,\lfloor j/8\rfloor,2}, \boldsymbol{x}_{\lfloor i/16\rfloor,\lfloor j/16\rfloor,1}\} \tag{36}$$

(see **Figure 1C**), so that each multiscale visual field owns 15 coefficients only (to be compared with the 784 pixels of the original image). **Figure 1D** displays a reconstructed image from the 4 central viewpoints at coordinates (7, 7), (7, 8) (8, 7) and (8, 8).

3.2.1.2. Algorithms

A generic sequential scene decoding setup is provided in algorithms 1 and 2. A significant algorithmic add-on when compared with formula (8) is the use of a *dynamic actions set* : $\mathcal{U}$. At each turn, the new selected action $\tilde{u}$ is drawn off from $\mathcal{U}$, so that the next choice is made over fresh directions that have not yet been explored. This implements the inhibition of return principle stated in Itti and Koch (2001). A second algorithmic add-on is the

[3]http://yann.lecun.com/exdb/mnist

TABLE 1 | Action selection metrics summary.

Name	Value	Equation #	Interpretation
Infomax	$H(q^{(n-1)}) - H(q^{(n)})$	(9)	Posterior mutual information maximization
Saliency	$\mathrm{KL}(q^{(n)}\|\|q^{(n-1)})$	(10)	Posterior *in*consistency maximization
VFE	$-\log p(\boldsymbol{x}^{(n)}\|\boldsymbol{u}^{(n)}; q^{(n-1)}) + \mathrm{KL}(q^{(n-1)}\|\|q^{(n)})$	(11)	Prior *in*consistency minimization
Compression Improvement	$\mathbb{E}_{\boldsymbol{z}\sim q^{(n-1)}}\left[-\log p(\boldsymbol{x}^{(n-1)}\|\boldsymbol{z}, \boldsymbol{u}^{(n-1)})\right] + \mathrm{KL}(q^{(n-1)}\|\|q^{(n-2)})$ $+\mathbb{E}_{\boldsymbol{z}\sim q^{(n)}}\left[\log p(\boldsymbol{x}^{(n-1)}\|\boldsymbol{z}, \boldsymbol{u}^{(n-1)})\right] - \mathrm{KL}(q^{(n)}\|\|q^{(n;n-2)})$	(26)	(approximate) Information Gain maximization
IGLB	$\mathbb{E}_{z\sim q^{(n-1)}}\left[\log p(\boldsymbol{z}\|\boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)})\right] + c$	(30)	(pessimistic) Information Gain maximization
IGUB	$\mathbb{E}_{z\sim q^{(n)}}\left[\log p(\boldsymbol{z}\|\boldsymbol{x}^{(n)}, \boldsymbol{u}^{(n)})\right] + c$	(35)	(optimistic) Information Gain maximization

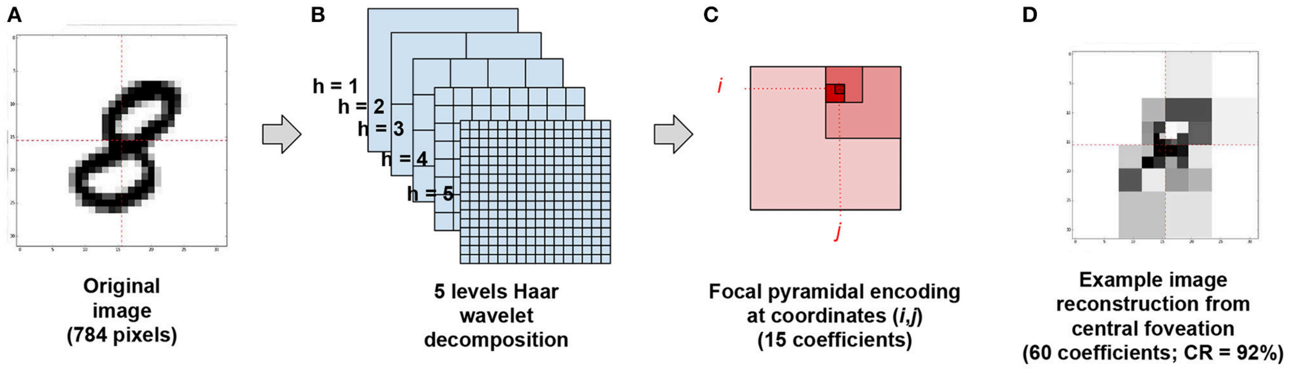

A
Original image (784 pixels)
B
h = 1
h = 2
h = 3
h = 4
h = 5
5 levels Haar wavelet decomposition
C
i
j
Focal pyramidal encoding at coordinates (i,j) (15 coefficients)
D
Example image reconstruction from central foveation (60 coefficients; CR = 92%)

FIGURE 1 | A–C Foveal "pyramidal" encoding from image. **(A)** An original MNIST sample is recast on a 32 × 32 grid. **(B)** It is then decomposed in a five levels Haar wavelets decomposition issuing a total of 1024 wavelet coefficient. **(C)** Then, for each gaze orientation $(i,j) \in \{0, ..., 15\}^2$, 3 × 5 wavelet coefficients are read out at coordinates (i,j), $(\lfloor \frac{i}{2} \rfloor, \lfloor \frac{j}{2} \rfloor)$, $(\lfloor \frac{i}{4} \rfloor, \lfloor \frac{j}{4} \rfloor)$, $(\lfloor \frac{i}{8} \rfloor, \lfloor \frac{j}{8} \rfloor)$ and $(\lfloor \frac{i}{16} \rfloor, \lfloor \frac{j}{16} \rfloor)$ in level descending order. **(D)** Example image reconstruction from reading 60 central coefficients at coordinates (7,7), (7,8), (8,7) and (8,8), issuing a 92% compression rate.

Algorithm 1 Prediction-Based Policy

```
Require: $p$ (emission density), $\rho$ (prior), $A$ (objective function), $\mathcal{U}$ (actions set)
  for $\boldsymbol{z}, \boldsymbol{u} \in \mathcal{Z}, \mathcal{U}$ do
    predict: $\tilde{\boldsymbol{x}}_{z,u} \sim p(X|\boldsymbol{z}, \boldsymbol{u})$
    $r(\boldsymbol{z}, \boldsymbol{u}) \leftarrow A(\boldsymbol{z}, \boldsymbol{u}, \tilde{\boldsymbol{x}}_{z,u}, p, \rho)$
  end for
  return $\tilde{\boldsymbol{u}} = \underset{\boldsymbol{u} \in \mathcal{U}}{\text{argmax}} \langle \rho, r(:, \boldsymbol{u}) \rangle$
```

Algorithm 2 Scene Exploration

```
Require: $p$ (emission density), $\rho_0$ (initial prior), $A$ (objective), $\mathcal{U}$ (actions set)
  $\rho \leftarrow \rho_0$
  while $H(\rho) > H_{\text{ref}}$ do
    choose: $\tilde{\boldsymbol{u}} \leftarrow$ Prediction-Based Policy($p, \rho, A, \mathcal{U}$)
    read: $\boldsymbol{x}_{\tilde{u}}$
    update: odd $\leftarrow \log p(\boldsymbol{x}_{\tilde{u}}|Z, \tilde{\boldsymbol{u}}) + \log \rho$
    $\rho \leftarrow$ softmax(odd) {the posterior becomes the prior of the next turn}
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{\tilde{\boldsymbol{u}}\}$
  end while
```

use of a threshold H_{ref} to stop the evidence accumulation process when enough evidence has been gathered. This threshold is a free parameter of the algorithm that sets whether we privilege a conservative (tight) or optimistic (loose) threshold. The stopping criterion needs to be optimized to arbitrate between resource saving and decoding accuracy.

The actual saccade exploration algorithm moreover adapts algorithm 2 the following way. The process starts from a loose assumption based on reading the root wavelet coefficient of the image, from which an initial guess ρ_0 is formed. Then, each follow-up saccade is defined as the gaze end-orientation $(i,j) \in [0,.., 15]^2$. The posterior calculation rests on up to 5 coefficient triplets (see Equation 36). After selecting gaze orientation (i,j), all the corresponding coordinates (i,j,h)'s are discarded from $\mathcal{U}$ and can not be reused for upcoming posterior estimation (for the final posterior estimate may be consistent with a uniform scan over the wavelet coefficients).

3.2.1.3. Baseline generative model

A generative model is learned for each $\boldsymbol{u} = (i,j,h)$ (making a total of 266 data models) over the 55,000 examples of the MNIST training set. For each category $\boldsymbol{z}$ and each viewpoint $\boldsymbol{u}$, a generative emission model is built over parameter set $\Theta_{z,u} = (\rho_{z,u}, \boldsymbol{\mu}_{z,u}, \boldsymbol{\Sigma}_{z,u})$,
so that:

$$\forall \boldsymbol{z}, \boldsymbol{u}, \tilde{\boldsymbol{x}}_{z,u} \sim p(X|\boldsymbol{z}, \boldsymbol{u}) = \mathcal{B}(\rho_{z,u}) \times \mathcal{N}(\boldsymbol{\mu}_{z,u}, \boldsymbol{\Sigma}_{z,u}) \quad (37)$$

with $\mathcal{B}$ a Bernouilli distribution and $\mathcal{N}$ a multivariate Gaussian. The role of the Bernouilli is to "gate" the multivariate Gaussian model in the high frequencies, where digit deformations is reflected in an alternating presence or absence of pixels for high level coefficients and at the periphery, allowing to discard the "white" triplets from the Gaussian moments calculation. Each resulting emission density $p(X|Z, \boldsymbol{u})$ is a mixture of Bernouilli-gated Gaussians over the 10 MNIST labels. On the inference side, the posterior is explicitly calculated using Bayes rule, i.e., $q(Z|\boldsymbol{x}, \boldsymbol{u}) = \text{softmax} \log p(\boldsymbol{x}|Z, \boldsymbol{u})$, issuing about 92% recognition rate on the MNIST test set when combining the 266 log likelihoods of each wavelet triplet of the full images with Equation (5), a typical recognition rate for shallow models.

3.2.2. Metrics Comparison

3.2.2.1. Baseline model and decoding compression

Different examples of a sequential scene decoding are presented in **Figure 2** for one MNIST sample using algorithm 2 and different objective functions. Note that several coefficient triplets are read at each end-effector position (i,j) (see **Figure 1C**). There is for instance a total of 5 triplets read out at the initial gaze orientation, and between 1 and 4 triplets read-out for each continuing saccade (not shown).

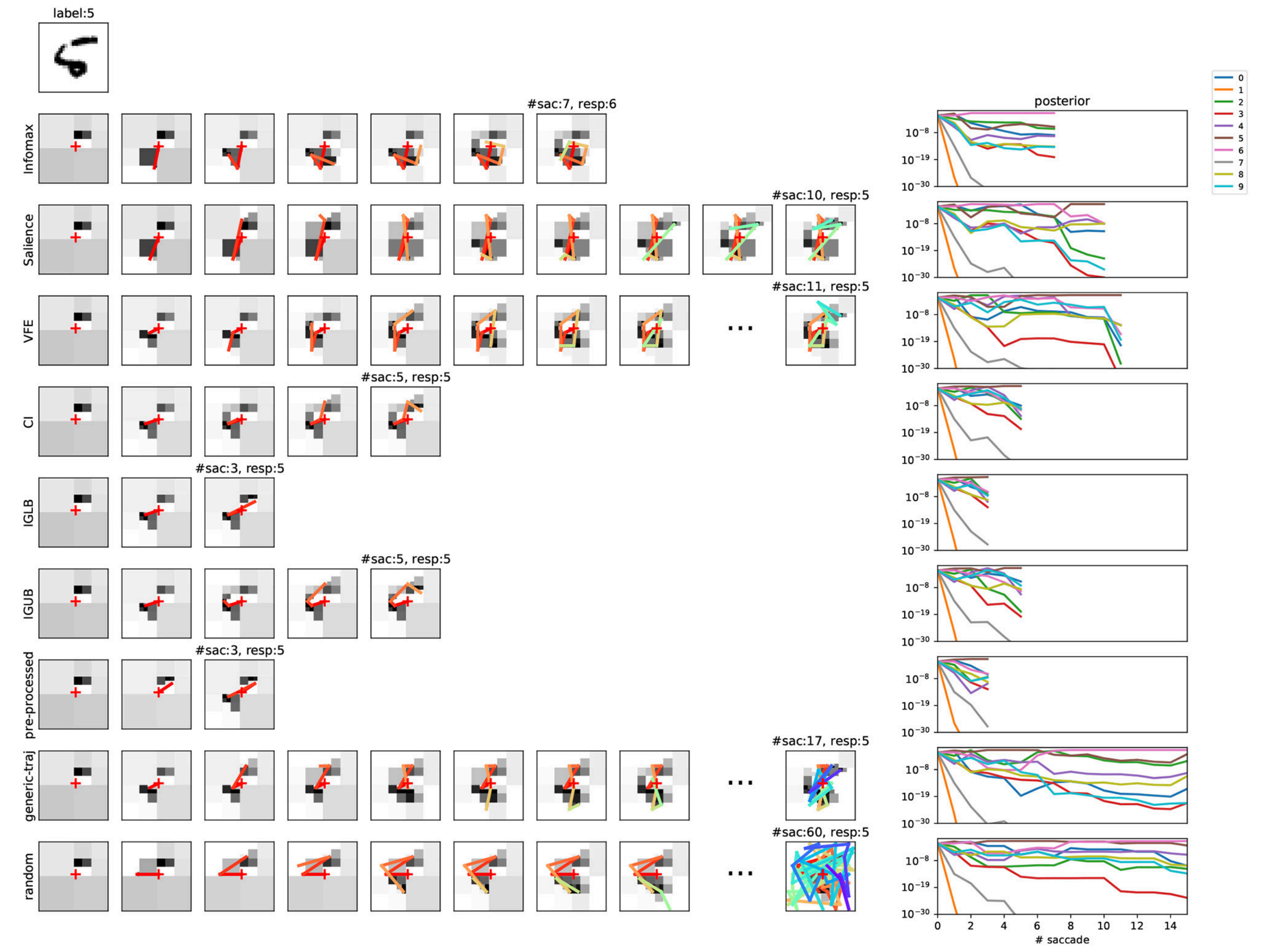


FIGURE 2 | Scene exploration through saccades in the foveated vision model. **(Top left)** Original MNIST sample to be decoded, with corresponding label. **(Left panel)** Course of saccades for different action selection metrics. Leftmost is the metric name. For each row, the number of thumbnail images reflects the number of saccades. The scene decoding reads from left to right: more wavelet coefficients are grabbed at each step, visually reflected in an increased reconstruction neatness. On overlay is the corresponding history of visual fixations (with rainbow time color code). The total number of saccades can vary for the different policies. Over the final thumbnail is the number of saccades and the final response. All decoding steps are shown except when $n > 10$. **(Right panel)** corresponding posterior update in function of the number of decoding steps, for $0 \leq n \leq 15$ (y-logarithmic scale, one color per competing label). Baseline model (Equation 37); $H_{\text{ref}} = 10^{-4}$.

Last, the *decoding compression rate* is defined as the proportion of wavelet coefficients that are bypassed for reaching decision. In **Figure 2** first row for instance, a total of 25 coefficient triplets is actually read-out from 7 saccades (not shown), representing about 10% of the total 256 coefficient triplets, issuing a 90% compression rate. The left-hand side of **Figure 3** shows how the classification rates vary in function of the average compression rate, for different objective functions and recognition threshold $H_{\text{ref}} \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. The objectives are also compared with a random baseline policy. The classification rates monotonically increase with a *decreasing* recognition threshold. Considering 92% as the upper bound here, a near optimal recognition rate is obtained at $H_{\text{ref}} = 10^{-5}$ for the CI objective. Though all objectives functions show a consistent increase of the classification rate with decreasing H_{ref}, the CI-based policy here overtakes the others policies. The Infomax and the VFE-based policies behave in a close-by fashion, and then the Salience-based policy provides a less effective scene decoding. All scene decoding policies provide elevated compression rates, with a close to optimal classification obtained at around 85% compression of the original data. It must be noticed that still a correct 90% classification rate can be obtained with a random policy at around 70% compression rate, reflecting a strong redundancy in the original images.

3.2.2.2. *Convolutional neural network*

A convolutional neural network (CNN) was designed in order to provide a more effective inference and facilitate comparison with state-of-the-art classifiers (see **Figure 4**). It is made of five convolution layers having each a distinct input corresponding to the five hierarchical levels of the wavelet decomposition. The CNN is biasless, uses a (2,2) stride for the convolutions

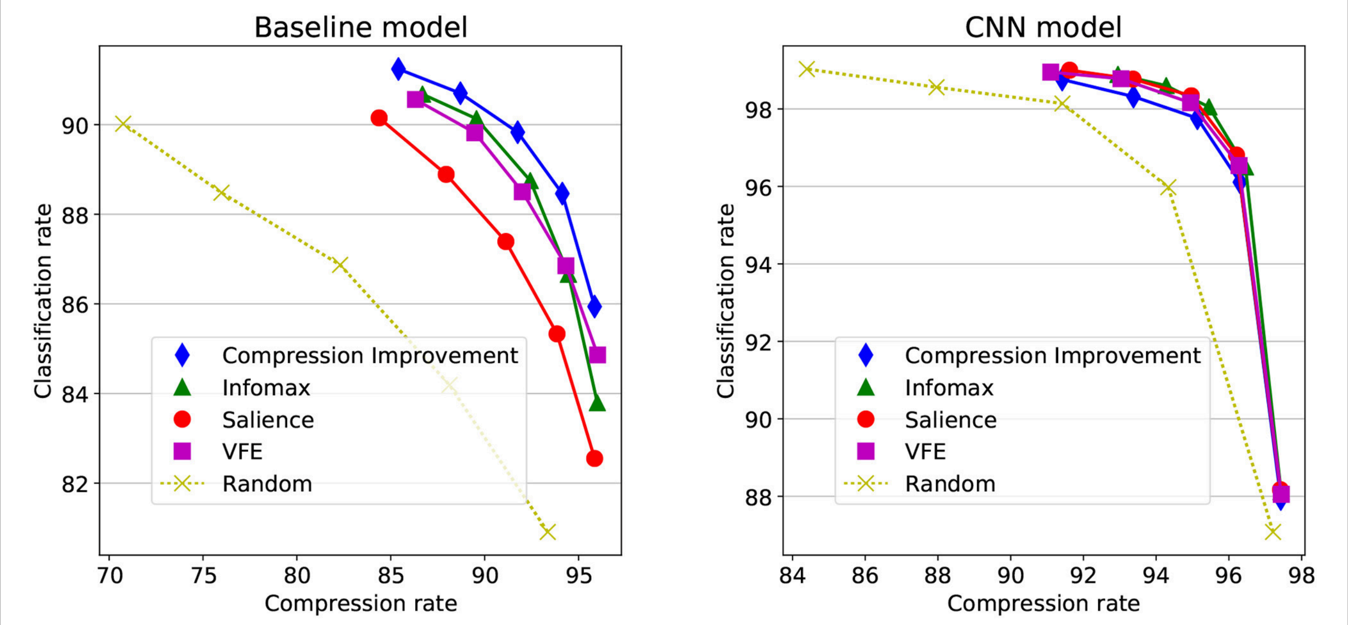


FIGURE 3 | Objective functions comparison. The classification rates and average compression rates are processed after 10,000 sequential scene decoding sessions on the MNIST test set, with different objective (or loss) functions and different values of H_{ref}. The classification rate is shown in function of the decoding compression rate. **(Left)** Baseline model, with recognition threshold H_{ref} varying from 10^{-5} up to $H_{\text{ref}} = 10^{-1}$ (from left to right). **(Right)** CNN model, with recognition threshold varying from $H_{\text{ref}} = 10^{-2}$ up to $H_{\text{ref}} = 1$ (from left to right).

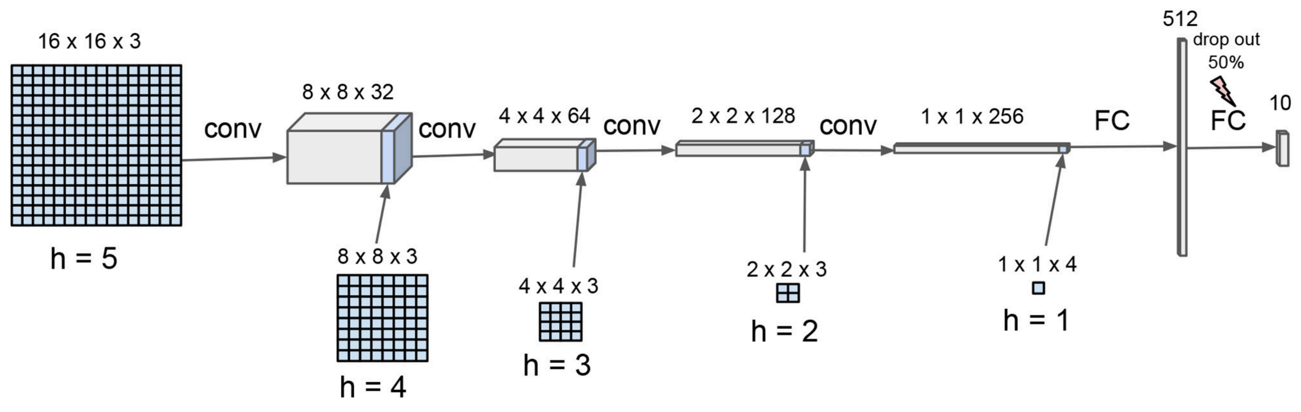


FIGURE 4 | Hierarchical convolutional neural network for scene decoding. The CNN is composed of four convolutional layers and one fully connected layer. The input wavelet hierarchical organization is reflected in scale-dependent input inlay, consistently with stride-2 convolutional spatial integration.

without max-pooling, promoting *neighbour independence* in the convolutional computation track. Rectified Linear Units are used in all layers, except for the final layer owning linear units.

The network was trained during about 10^6 epochs with Tensorflow[4] on a laptop. Sparse foveal-consistent inputs are used for the training. For each training example, many gaze orientation $(i_1, j_1), ..., (i_n, j_n)$ are chosen at random, mimicking a n-views foveal visual scan, with n randomly set from interval $\{1, .., 256\}$. The multi-level input maps are then fed with the corresponding wavelet coefficients triplets, "pyramidally" distilled from $h = 5$ to $h = 0$. The final network is expected to perform recognition on *randomly compressed* images, for which some wavelet coefficients are kept and some wavelet coefficients are discarded. With standard parameter tuning [Adam optimizer – Kingma and Ba (2014) – with a learning rate equal to 10^{-4}], the network attains a 99% recognition rate on the test set with non-compressed wavelet transformed inputs (full information case).

The cross-entropy loss used in training allows to interpret the network output as approaching the data log-likelihood (up to a constant), i.e., $\text{CNN}(\boldsymbol{x}_{i,j}) \simeq \log p(\boldsymbol{x}_{i,j}|Z, (i,j)) + c$. For decoding a scene, the input layers are initialized at zero

[4]https://www.tensorflow.org

and progressively filled with new wavelet coefficients obtained during the scene exploration. The output is updated by adding supplementary data at the input only, complementing the data that was previously read, with $\exp\left[\text{CNN}(\boldsymbol{x}^{(1:n)}|\boldsymbol{u}^{(1:n)})\right] \propto p(\boldsymbol{x}^{(1:n)}|Z, \boldsymbol{u}^{(1:n)}) \propto p(Z|\boldsymbol{x}^{(1:n)}, \boldsymbol{u}^{(1:n)}) = q^{(n)}$. The posterior update is thus implemented *from the data*. There is no recurrence, sequential accumulation or memory implemented in the network (like in Equation 6).

Following algorithm 2 with the CNN as the approximate cumulated posterior estimator, the decoding efficacy is shown for different objective functions on the right-hand side of **Figure 3**, with $H_{\text{ref}} \in \{1, 0.3, 0.1, 0.03, 0.01\}$. It is to be noted the CNN is only used for estimating the $q^{(n)}$'s posterior distributions, with the baseline Bernouilli-gated multivariate Gaussian model (Equation 37) used on the predictive/generative side. A clear decoding improvement is obtained when compared with the left-hand side of **Figure 3**, with higher classification rates with less signal, attaining about 98,8% correct classification with less than 8% of the original image. Still, the general good performances of the decoder blurs the differences between the different policies. All objectives appear here equally good at effectively decoding the scene (except for the random action selection policy).

3.2.2.3. Faulty model and failure robustness

Predictive policies are known to heavily rest on the generative models, that makes them sensible to model flaws. Resistance to model flaws is thus a property that should be prioritized when acting in unknown or coarsely modeled environments, or in the course of learning. In contrast with CNN-based optimal decoding, a failed probabilistic model was designed by simply setting $\rho_{u,z} = 1$ in Equation (37). This tends to overestimate the signal strength at high frequencies, predicting a dense signal in effectively sparse regions. The classification accuracies are presented on **Figure 5A** for the different objective functions considered here. In complement to the Compression Improvement (Equation 26), the two variants referred as the IG Lower Bound (Equation 30) and the IG Upper Bound (Equation 35) are also considered.

The faulty model allows here to nicely separate the different metrics with regards to their optimistic vs. conservative flavor. While the CI is here barely better than a random sampling, its conservative and optimistic variants, respectively, do clearly better and clearly worst than random exploration. The VFE loss and the Saliency objectives, as expected, amplify this effect with a strong robustness to model flaws for the VFE loss and, at reverse, a strong sensitivity to model flaws for the Saliency objective. The Infomax also falls here in the optimistic category for its blindness to sequential consistency makes it update the posterior the wrong way.

3.2.3. Scaling Up IG Computation

The scaling of the model needs to be addressed when large control spaces are considered. All predictive policies rely on a mixed encoding setup that implies to consider all $\boldsymbol{u}$'s and all $\boldsymbol{z}$'s in the prediction, which scales like $O(|\mathcal{U}| \times |\mathcal{Z}|^2)$ when the predicted posterior is needed in the objective/loss calculation, which is the case for the Infomax, the Saliency, the VFE the CI and the IGUB (algorithm 1), and $O(|\mathcal{U}| \times |\mathcal{Z}|)$ in the IGLB case for it can bypass the post-sample posterior calculation. A quadratic cost may still be considered too heavy in real-case applications, implying to consider cheaper setups.

- A first simplification, referred as the "sharp" IGLB in **Figures 5B,C**, only samples a single $\boldsymbol{z}$ from $q^{(n-1)}$, i.e.

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}}{\text{argmax}}\, q^{(n-1)}(\boldsymbol{z})$$

  with $\hat{\boldsymbol{z}}$ the current *guess*, making the predictive policy scale like $O(|\mathcal{U}|)$ for the main loop of algorithm 1 is now over the $\boldsymbol{u}$'s only.
- An additional simplification can be obtained when considering the IGLB objective alone (Equation 30), for it is, on contrary to all other objectives, independent of the context $q^{(n-1)}$. For a given model p, and for every guess $\hat{\boldsymbol{z}} \in \mathcal{Z}$, all the predictive log posteriors $\log p(\hat{\boldsymbol{z}}|\hat{\boldsymbol{x}}_{\hat{z},u}, \boldsymbol{u})$ can be pre-processed using the *mode* of the predicted visual field $\hat{\boldsymbol{x}}_{\hat{z},u} = \underset{\boldsymbol{x}}{\text{argmax}}\, p(\boldsymbol{x}|\hat{\boldsymbol{z}}, \boldsymbol{u})$ as a sample. This results in a set of class-consistent *action maps* providing, for each $\boldsymbol{u} \in \mathcal{U}$, the expected log posterior value $\log p(\hat{\boldsymbol{z}}|\hat{\boldsymbol{x}}_{\hat{z},u}, \boldsymbol{u})$, given $\hat{\boldsymbol{z}}$ (**Figure 6**, first row). Then, for each guess $\hat{\boldsymbol{z}}$, a class-specific visual exploration strategy can be *pre-processed*, following a log-posterior descending order over the $\boldsymbol{u}'s$, from higher IGLB values of toward lower IGLB values (brownish toward whitish in **Figure 6**). $|\mathcal{Z}|$ saccade trajectories of size $|\mathcal{U}|$ are then calculated offline and stored in ordered lists, with a $O(|\mathcal{U}| \times |\mathcal{Z}|)$ memory load but only $\simeq O(1)$ readout computational cost. In practice, the viewpoint selected at step n depends on the current guess $\hat{\boldsymbol{z}}$, with on-the-fly trajectory switch if the guess is revised during the course of saccades. This strategy is referred a the *pre-processed trajectories* in **Figures 2**, **5B,C**.
- For comparison, a *generic* trajectory was also computed using

$$\overline{\text{IGLB}}(\boldsymbol{u}) = \mathbb{E}_{z\sim p(Z)}\left[\log p(\boldsymbol{z}|\hat{\boldsymbol{x}}_{z,u}, \boldsymbol{u})\right] \tag{38}$$

  with a uniform prior over the $\boldsymbol{z}'s$, i.e., $p(\boldsymbol{z}) = \frac{1}{|\mathcal{Z}|}$. It is referred a the *generic trajectory* in **Figures 2**, **5B,C**.This strategy is useful in the absence of a definite guess (uniform initial prior for instance).

The action maps allow to analyze in detail the class-consistent orientations (that appear brownish) as opposed to the class-inconsistent orientations (pale orange to white). First to be noticed is the relative scarceness of the class-consistent orientations. A small set of saccades is expected to provide most of the classification information while the rest of the image is putatively uninformative (or even misleading if whitish). A second aspect is that the class-relevant locations are all located in the central part of the images, so there is very few chance for the saccades to explore the periphery of the image where little information is expected to be found. This indicates that the model has captured the essential concentration of class-relevant

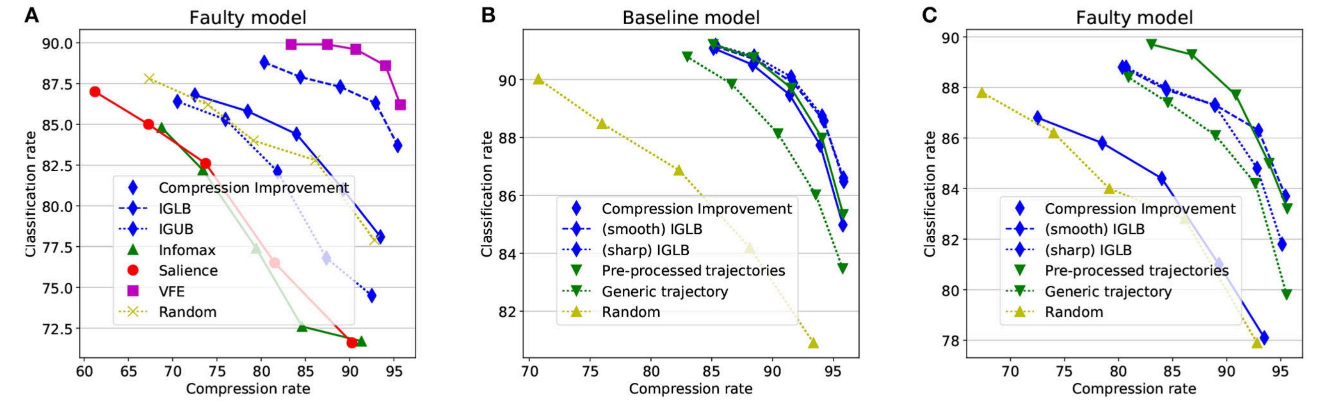


FIGURE 5 | Method comparisons. **(A)** Objective functions comparison in a faulty model (see **Figure 3**). **(B)** Information Gain-based computational schemes compared on the baseline model. **(C)** Information Gain-based computational schemes compared on the faulty model. The recognition threshold varies from $H_{\text{ref}} = 10^{-5}$ up to $H_{\text{ref}} = 10^{-1}$ (from left to right).

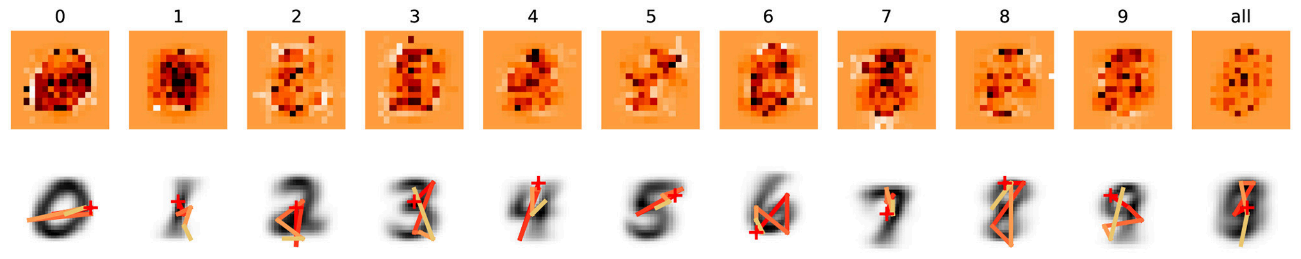


FIGURE 6 | Action maps and pre-processed trajectories. **Upper row** (except rightmost map) color-coded pre-processed guess-consistent action maps for $\hat{z} \in \{0, .., 9\}$, and $\boldsymbol{u} \in \{0, .., 15\}^2$, using the baseline generative model, from low (whitish) to high (brownish) log posteriors. **Upper row, rightmost map:** Class-independent expected IGLB map. **Lower row:** Corresponding pre-processed visual scan-path (the red "+" provides the initial gaze orientation). Only the 5 first saccades are shown, with average class prototype in the background. The rightmost background image is the average over all classes.

information in the central part of the images for that particular training set.

The different simplification strategies are compared in **Figures 5B,C** over the baseline and the faulty models. Both the sharp IGLB and the pre-processed trajectories are shown consistent with the CI objective on **Figure 5B**, despite their considerably lower computational cost, while the generic trajectory strategy appears less effective. Interestingly, those computational simplifications also remain valid when robustness to model flaws is considered (**Figure 5C**). Both the sharp IGLB and the pre-processed trajectories allow to reach both robustness and effective classification rates at considerably lower cost than the "smooth" IGLB.

4. CONCLUSION

Stemming from the fovea-based scene decoding problem, a generic *predictive* action selection framework was presented which, accordingly with (Najemnik and Geisler, 2005), rests on a predictive accuracy metric to choose action. An "active" inference approach is also considered, which, accordingly with Friston et al. (2012), optimizes sensory samples selection through action. In our case, the visual field is interpreted under a *mixed emission* model for the visual data is both generated by the viewpoint and the scene constituents. This allows to unify the many objective functions proposed in the literature under a single metric referred as the Compression Improvement (CI) in Schmidhuber (2007), that is shown to provide a consistent interpretation for most of the objective functions used in perception-driven control.

Two variants of the CI objective are then proposed, using either the pre-sample or the post-sample posterior in the approximation. In the pre-sample case, it is shown to be an Information Gain Lower Bound objective that always underestimate the actual Information Gain. The IGLB objective is said conservative for it should prevent from searching for conflicting visual data that may challenge the current interpretation. On the other hand, it is expected to lower the risk of failed interpretation in the case of a (erroneous) conflicting predictions. Conversely, in the post-sample case, the approximate CI is shown to always *overestimate* the actual Information Gain, making it the Information Gain Upper Bound objective (IGUB)—or "optimistic" IG objective. Following the IGUB is expected to perform a more thorough scene exploration for it may preferentially head toward conflicting visual data that may challenge the current interpretation. On the other hand, it is also expected to increase the risk of failed interpretation in the case of (erroneous) conflicting predictions.

Remarkable is the fact that both the Saliency objective, the Infomax objective and the Free Energy epistemic cost, that are classic metrics of the literature, show a consistent inclusion in a more general approximate Information Gain maximization principle. Using for instance the Variational Free Energy (Friston et al., 2015) as a loss (instead of the IGLB) is expected to *bias* the action selection in an even more conservative way. Conversely, using the Saliency objective (Itti and Baldi, 2005) instead of the IGUB is expected to *bias* the action selection in an even more optimistic way, subsequently increasing the risk of a failed scene interpretation.

The presented numerical experiments thus highlight different aspects of the setup. A first and principal result is that state-of-the-art recognition rates are obtained with sequential fovea-based computation using less than 10% of the original signal. This strong input compression is made possible for the visual data owns lot of redundancies that are not used at their best in computer vision, doing useless computations over large parts of the visual scene. The satisfactory results obtained in that case reflect the advantage of mixing a predictive controller with accurate state-of-the-art predictors, here a deep neural network.

A second result is the sub-optimality of many action selection metrics used in literature, like the "Infomax" (Butko and Movellan, 2010) and the "Salience" objectives (Itti and Baldi, 2005), when the scene decoding setup is considered. Their sub-optimality is not manifest with finely-tuned generative models, but becomes patent when a coarse of faulty model is used. This may appear counter-intuitive at first sight for the Infomax objective is vastly dominant in predictive control (Najemnik and Geisler, 2009), while the Salience objective provides among the best predictions for human fixation zones (Itti and Baldi, 2005). The mixed performances of the Salience objective in predictive control may however be attenuated when learning is considered. Heading toward inconsistently modeled places is indeed a sensible behavior when the model is immature. This entails maximizing predictions errors, which is a relevant principle long considered in sparse reinforcement learning (Schmidhuber, 1991; Oudeyer and Kaplan, 2008; Pathak et al., 2017). This trade-off reflects a more general contradiction between exploiting at best the current knowledge from past observations vs. challenging the current interpretation to leverage conflicting facts, a variant of the exploration/exploitation trade-off.

Last, a notorious drawback of the predictive setup is its computational cost scaling with the size of the actions sets, that may grow combinatorially fast with increasing degrees of freedom. Real-world predictive control is thus in need for computationally-effective predictive models, here attainable with the Information Gain Lower Bound (IGLB) objective, that, though maximizing the Information Gain in approximation, allow for low-complexity calculation when replacing the exact posterior with a single guess in the prediction. In discrete latent spaces, it is thus possible to pre-process guess-specific offline trajectories, allowing to bypass computationally-demanding predictions. This strongly simplified setup is shown efficient in our case, showing both competitive decoding compression rates and good robustness to model flaws.

IG-driven fovea-based sequential processing may finally be useful in the case of high dimensional input data (like in e.g., computer vision), and should be tested on more challenging computer vision setups. It is also to be determined how far IG-based action selection may extend to more general partially observed environments, and whether they could challenge more established actions selection strategies in open-ended control setups.

AUTHOR CONTRIBUTIONS

The author confirms being the sole contributor of this work and has approved it for publication.

ACKNOWLEDGMENTS

The author is grateful to Laurent Perrinet and Anna Montagnini, from Institut de Neurosciences de la Timone, for inspiring ideas, valuable comments and human vision literature guidance. He would also like to thank the reviewers for their valuable comments and suggestions to improve the article.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot.2018.00076/full#supplementary-material

REFERENCES

Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. *Int. J. Comput. vis.* 1, 333–356. doi: 10.1007/BF00133571

Bajcsy, R. (1988). Active perception. *Proc. IEEE* 76, 966–1005. doi: 10.1109/5.5968

Baum, L. E., and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Stat.* 37, 1554–1563. doi: 10.1214/aoms/1177699147

Butko, N. J., and Movellan, J. R. (2010). Infomax control of eye movements. *IEEE Trans. Auton. Ment. Dev.* 2, 91–107. doi: 10.1109/TAMD.2010.2051029

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). "A recurrent latent variable model for sequential data," in *Advances in Neural Information Processing Systems* (Montreal, QC), 2980–2988.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. (2016). "Sequential neural models with stochastic layers," in *Advances in Neural Information Processing Systems* (Barcelona), 2199–2207.

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nat. Rev. Neurosci.* 11, 127–138. doi: 10.1038/nrn2787

Friston, K., Adams, R., Perrinet, L., and Breakspear, M. (2012). Perceptions as hypotheses: saccades as experiments. *Front. Psychol.* 3:151. doi: 10.3389/fpsyg.2012.00151

Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., and Pezzulo, G. (2017). Active inference: a process theory. *Neural Comput.* 29, 1–49. doi: 10.1162/NECO_a_00912

Friston, K., Rigoli, F., Ognibene, D., Mathys, C., Fitzgerald, T., and Pezzulo, G. (2015). Active inference and epistemic value. *Cogn. Neurosci.* 6, 187–214. doi: 10.1080/17588928.2015.1020053

Guo, C., and Zhang, L. (2010). A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression. *IEEE Trans. Image Process.* 19, 185–198. doi: 10.1109/TIP.2009.2030969

Hinton, G. E., and Zemel, R. S. (1994). "Autoencoders, minimum description length and helmholtz free energy," in *Advances in Neural Information Processing Systems* (Denver, CO), 3–10.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). "Vime: variational information maximizing exploration," in *Advances in Neural Information Processing Systems* (Barcelona), 1109–1117.

Itti, L., and Baldi, P. (2005). "Bayesian surprise attracts human attention," in *Advances in Neural Information Processing Systems* (Vancouver, BC), 547–554.

Itti, L., and Koch, C. (2000). A saliency-based search mechanism for overt and covert shifts of visual attention. *Vis. Res.* 40, 1489–1506. doi: 10.1016/S0042-6989(99)00163-7

Itti, L., and Koch, C. (2001). Computational modelling of visual attention. *Nat. Rev. Neurosci.* 2, 194–203. doi: 10.1038/35058500

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *J. Fluids Eng.* 82, 35–45. doi: 10.1115/1.3662552

Kingma, D. P., and Ba, J. (2014). Adam: a method for stochastic optimization. arXiv:1412.6980.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). "Empowerment: a universal agent-centric measure of control," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, Vol. 1 (Edinburgh: IEEE), 128–135.

Kortum, P., and Geisler, W. S. (1996). Implementation of a foveated image coding system for image bandwidth reduction. *Hum. Vis. Electr. Imaging*, 2657, 350–360. doi: 10.1117/12.238732

Mohamed, S., and Rezende, D. J. (2015). "Variational information maximisation for intrinsically motivated reinforcement learning," in *Advances in Neural Information Processing Systems* (Montreal, BC), 2125–2133.

Mussa-Ivaldi, F. A., and Solla, S. A. (2004). Neural primitives for motion control. *IEEE J. Ocean. Eng.* 29, 640–650. doi: 10.1109/JOE.2004.833102

Najemnik, J., and Geisler, W. S. (2005). Optimal eye movement strategies in visual search. *Nature* 434, 387–391. doi: 10.1038/nature03390

Najemnik, J., and Geisler, W. S. (2009). Simple summation rule for optimal fixation selection in visual search. *Vis. Res.* 49, 1286–1294. doi: 10.1016/j.visres.2008.12.005

Osterberg, G. (1935). Topography of the layer of the rods and cones in the human retima. *Acta ophthalmol.* 13, 1–102.

Oudeyer, P.-Y., and Kaplan, F. (2008). "How can we define intrinsic motivation?," in *Proceedings of the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Lund University Cognitive Studies, Lund: LUCS, Brighton*. Lund; Brighton: Lund University Cognitive Studies, LUCS.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning (ICML)*, Vol. 2017, (Sydney, NSW).

Potthast, C., Breitenmoser, A., Sha, F., and Sukhatme, G. S. (2016). Active multi-view object recognition: a unifying view on online feature selection and view planning. *Robot. Auton. Syst.* 84, 31–47. doi: 10.1016/j.robot.2016.06.013

Rao, R. P., and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.* 2, 79–87. doi: 10.1038/4580

Schmidhuber, J. (1991). "Curious model-building control systems," in *Neural Networks, 1991. 1991 IEEE International Joint Conference on* (Singapore: IEEE), 1458–1463.

Schmidhuber, J. (2007). "Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity & creativity," in *International Conference on Discovery Science* (Sendai: Springer), 26–38.

Schmidhuber, J., and Huber, R. (1991). Learning to generate artificial fovea trajectories for target detection. *Int. J. Neural Syst.* 2, 125–134. doi: 10.1142/S012906579100011X

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, Vol. 1. Cambridge: MIT Press.

Tishby, N., and Polani, D. (2011). Information theory of decisions and actions. in *Perception-action Cycle*, eds V. Cutsuridis, A. Hussain, and J. G. Taylor (Boston, MA: Springer), 601–636.

van Beers, R. J., Haggard, P., and Wolpert, D. M. (2004). The role of execution noise in movement variability. *J. Neurophysiol.* 91, 1050–1063. doi: 10.1152/jn.00652.2003

Wald, A. (1945). Sequential tests of statistical hypotheses. *Ann. Math. Stat.* 16, 117–186. doi: 10.1214/aoms/1177731118

Wang, Z., Lu, L., and Bovik, A. C. (2003). Foveation scalable video coding with automatic fixation selection. *IEEE Trans. Image Process.* 12, 243–254. doi: 10.1109/TIP.2003.809015

Yarbus, A. L. (1967). "Eye movements during perception of complex objects," in *Eye Movements and Vision*, ed A. L. Yarbus (Boston, MA: Springer), 171–211.

Conflict of Interest Statement: The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers
in Robotics and AI

REVIEW
published: 18 December 2018
doi: 10.3389/frobt.2018.00134

Toward Computational Motivation for Multi-Agent Systems and Swarms

Md Mohiuddin Khan, Kathryn Kasmarik and Michael Barlow*

School of Engineering and Information Technology, University of New South Wales, Canberra, ACT, Australia

Motivation is a crucial part of animal and human mental development, fostering competence, autonomy, and open-ended development. Motivational constructs have proved to be an integral part of explaining human and animal behavior. Computer scientists have proposed various computational models of motivation for artificial agents, with the aim of building artificial agents capable of autonomous goal generation. Multi-agent systems and swarm intelligence are natural extensions to the individual agent setting. However, there are only a few works that focus on motivation theories in multi-agent or swarm settings. In this study, we review current computational models of motivation settings, mechanisms, functions and evaluation methods and discuss how we can produce systems with new kinds of functions not possible using individual agents. We describe in detail this open area of research and the major research challenges it holds.

Keywords: intrinsic motivation, artificial intelligence, cognitive development, swarms, multi-agent systems, exploration, curiosity

OPEN ACCESS

Edited by:
Gianluca Baldassarre,
Italian National Research Council, Italy

Reviewed by:
Yara Khaluf,
Ghent University, Belgium
Heiko Hamann,
Universität zu Lübeck, Germany

****Correspondence:***
Md Mohiuddin Khan
mohiuddin.khan@adfa.edu.au

Specialty section:
This article was submitted to Computational Intelligence, a section of the journal Frontiers in Robotics and AI

Received: *30 May 2018*
Accepted: *03 December 2018*
Published: *18 December 2018*

Citation:
Khan MM, Kasmarik K and Barlow M (2018) Toward Computational Motivation for Multi-Agent Systems and Swarms. Front. Robot. AI 5:134. doi: 10.3389/frobt.2018.00134

INTRODUCTION

Artificial intelligence has come a long way toward developing intelligent systems. We are in an era where autonomous cars are on the verge of roaming the streets, chess programs can beat grandmasters, and handheld devices understand and translate speech in real-time. However, we are still far from developing artificial agents capable of demonstrating, human-like adaptive behavior, and open-ended learning. Research areas such as autonomous mental development (Thomaz and Breazeal, 2006) and developmental cognitive robotics (Asada et al., 2009) aim to address these challenges.

One of the important features of the artificial agents of the future will be their ability to gain knowledge and learn skills without explicit feedback from humans as well as adapt the behavior according to external and internal needs. As described in Russell and Norvig (2016), agent behavior can be guided by goals and utilities. In this paper we define motivation as a mechanism that generates goals as an intermediary between sensation and action selection in an agent. These “motivated” agents will use their acquired knowledge and skills to build increasingly complex repertoires of behaviors. Computational models of motivations have been proposed, drawing inspiration and insight from biological (Gatsoulis and Mcginnity, 2015), neural (Gottlieb et al., 2016), and evolutionary (Singh et al., 2010) perspectives. Computational models of motivation enables artificial agents to gather knowledge, seek competence, and select goals based on a combination of their individual experiences, preferences, and environmental characteristics (Merrick and Shafi, 2011). While the general concept of motivation is broad and has many facets, in our paper we focus on how motivation has been defined in artificial agents. Essentially a human and animal trait, researchers have used various notions, mathematical models, and frameworks to define motivation for artificial agents.

Inspired by insights from human psychology, these motivational models propose to incorporate open-ended learning, autonomous skill acquisition, and progressive learning in artificial agents. However, the current computational models of motivation have seldom been extended and explored in the social context- where multiple artificial agents exist, communicate and interact. Multi-agent systems and swarms are two examples of such contexts.

Multi-agent systems are a logical extension of the single agent idea. Studies involving multi-agent systems include the communication and behavior (Panait and Luke, 2005) among multiple artificial agents. With combined goals, actions, domain knowledge, and interactions (Stone and Veloso, 2000), these systems pose unique challenges that are absent in single agent settings. They can be robust and scalable, while introducing complexity in cooperation and communication, and capable of providing a platform to build social intelligence for artificial agents (Dautenhahn, 1995).

The term "Swarm intelligence" was first coined by Beni and Wang (1993). As they explained (Beni, 2004), a machine can be defined as "intelligent" if it demonstrates behavior which is "*neither random nor predictable.*" Following this definition, an intelligent swarm was defined as a group of non-intelligent agents that can collectively demonstrate intelligent behavior. Swarm intelligence is inspired by behavior occurring in insects, birds, fish and other organisms in nature. These biological systems have existed for millennia, and they have been efficiently solving complex problems through apparently simple rules. The current approach to swarm intelligence is a culmination of observations and findings from both biologists (Beekman et al., 2008) and computer scientists (Brambilla et al., 2013).

In this paper, we review the state of the art of the computational models of motivation and present a comprehensive review of this research. We talk about the various aspects of the current works and discuss the scope of extending them to multi-agent systems and swarms. This article has the following main contributions:

1. We review existing computational models of motivation, based on their setting, mechanism, function, and evaluation methods. This provides a structured overview of the existing research and a framework in which to introduce multi-agent systems and swarms to the study of computational motivation.
2. We characterize the open area of research that involves computational models of motivation in a social context, specifically, multi-agent systems, and swarms. We do this by examining motivation on the traditional axes of competence and knowledge, while introducing a new axis of social vs. individual agents.
3. Finally we determine the major challenges and benefits of designing computational motivation for multi-agent and swarm settings.

In the related works section, we discuss the existing surveys on computational models of motivation and justify the position of this survey. In the next section, we present the main contribution of this paper. It starts with a discussion on motivation and intrinsic motivation (IM), then provides a structural summary of the current works on computational models of intrinsic motivation. It concludes by presenting an outlook and summary on computational motivation in a social context. In the discussion section, we discuss the social side of intrinsic motivation from various aspects, present the major research challenges in that context, and conclude the paper.

RELATED WORK

A number of surveys have been produced in the area of computational motivation. This section discusses the focus of each of the existing surveys and justifies the need for a new survey studying computational motivation in a social context.

Oudeyer and Kaplan (2007) provided a typology of computational approaches to motivation. They assumed that any particular Computational Reinforcement Learning (CRL) framework could be used to realize motivation signals. Hence, their typology is based on the formal definition of the reward used in a framework. Characterizing a robot by having a number of sensor channels and motor channels, they classified motivation models into the following categories:

1. Knowledge-based: Knowledge-based systems use "measures that are related to the capacity of the system to model its environment" (Mirolli and Baldassarre, 2013b). The knowledge, for example, can be computed from the past sensorimotor values. Once the knowledge is acquired, the difference between the estimated knowledge and the actual perceived value can be used to design the reward. For example, the intrinsic reward can be proportional to the improvement of the prediction. In this case, the robot will be "intrinsically motivated" to maximize prediction progress, i.e., to minimize the prediction errors. In essence, knowledge-based models put the emphasis on how much an artificial agent "knows" about the environment.
2. Competence-based: In these models, the reward for intrinsic motivations is designed based on what an agent "can do" with respect to a particular goal or task. In these models, intrinsic rewards are associated with an agent's ability to reach a certain state or perform a certain activity. For example, a robot can be rewarded proportionally to the progress in learning a task, driving it toward goals that are rapidly improvable and deterring it from situations that are too difficult or too easy to gain enough competence.

In the survey section, we will use these categories as the baseline from which we introduce motivation in a social context. A related review (Oudeyer et al., 2007) describes a robot as having two modules—a learning machine and meta-learning machine. While the learning machine learns to predict the sensorimotor consequences of an executed action, the meta-learning machine learns to predict the errors of the learning machine. The prediction made by the meta-predictor is used as an intrinsic reward. The authors divided the existing approaches in three categories based on how these predictions are exploited to generate intrinsic motives. These are error maximization (Thrun, 1995; Huang and Weng, 2002; Barto et al., 2004; Marshall et al., 2004), progress maximization (Herrmann et al., 2000; Kaplan and Oudeyer, 2003), and similarity-based progress maximization (Schmidhuber, 1991). This survey provides an effective classification of the computational motivation models by distinguishing between learning and meta-learning with a focus on how agents can explore environments to gather effective information.

While the previously mentioned surveys discussed the mechanisms of intrinsic motivation, they do not provide a clear insight on the functional roles of motivation. (Mirolli and Baldassarre, 2013a,b) focus on the knowledge-based and competence-based models of intrinsic motivation from the functional aspect. They argued that "*the ultimate function of intrinsic motivation is to support the cumulative learning of skills rather than knowledge.*" They have analyzed some of the knowledge-based mechanism regarding their contribution to competence acquisition. They argue that to facilitate cumulative skill acquisition based on intrinsic motivation, one has to focus on the hierarchy and modularity of the skill organization framework. In this framework, knowledge-based reward signals can act at a lower level to learn the world-model, while competence-based signals can work as a selector deciding which skill is to learn.

Schmidhuber (2010) proposed a typology of intrinsic motivation from a different perspective. He pointed out that most of the intrinsically motivated systems have the following components:

1. A model/encoder/predictor that captures the sensory inputs, internal states, reinforcement signals and actions.
2. An intrinsic reward scheme that determines the learning progress of the model.
3. A reinforcement learner that maximizes the future expected reward.

Hence, this typology can be created considering the types of these components. A set of companion questions for each of the components is added and answering the questions can help build a detailed topology. Schmidhuber presented a general, theoretical framework that one can use to build a typology of intrinsically motivated systems. This typology is based on the consideration that intrinsic motivation will always be implemented through reinforcement signals.

Barto (2013) provides an overview of intrinsic motivation with regard to Reinforcement Learning (RL). He highlighted the suitability of RL to capture the principles of motivation in artificial systems by connecting drive theory with reward maximization. Barto pointed out that RL framework "*does not care*" where the reward signal is coming from. This makes it possible to introduce an "intrinsic reward" which would be generated from within the organism but won't affect the whole RL mechanism. Hence it can naturally accommodate intrinsic motivation. In addition to that, he highlighted that intrinsic reward signals can mimic the evolutionary success of organisms.

Computational models of motivations were surveyed as a part of computational value systems (Merrick, 2017). Value systems define behavioral responses of intelligent beings with regard to the external environment. The term "computational value systems" extends the idea toward artificial agents such as robots. A brief summary on the implementation of motivated reinforcement learning as value systems is provided. Merrick (Merrick, 2013) further reviews the existing novelty-based models of intrinsic motivation with a focus on building combined motivation models and integrated learning architecture.

In a more recent work (Roohi et al., 2018), application of intrinsic motivation in player modeling and game testing was reviewed. This work concludes that while a few parameters of intrinsic motivation are frequently implemented in the existing works, some important features are generally overlooked. They also point out the need for more complex motivational models and better ways to evaluate them. The purpose of our paper is to extend the existing views on intrinsic motivation beyond individual agents to multi-agent and swarm settings.

A summary of the existing surveys is presented in **Table 1**. It is evident from the table that there is no recent survey on computational intrinsic motivation that provides insight into its use in a social context. Till now, the most commonly referred to survey of intrinsic motivation is the typology provided by Oudeyer and Kaplan (2007). Their seminal work provides a comprehensive view of a formal framework for motivation. It provides a review of the then existing computational models of IM. Since their work in 2007, intrinsic motivation has become one of the most attractive research areas in cognitive science and autonomous mental development. Moreover, with the emergence of areas such as deep learning (Sigaud and Droniou, 2016; Wong, 2016) on autonomous mental development (Lake et al., 2017), computational models of motivation have become relevant in newer dimensions. In this survey, we categorize the existing approaches and argue the benefits of extending them to social settings- specifically multi-agent or swarm settings.

SURVEY: FROM MOTIVATION IN INDIVIDUAL AGENTS TO MOTIVATION IN SOCIAL SETTINGS

In the Motivation and Intrinsic Motivation from a Psychological Perspective section, we introduce motivation from a psychological perspective. The next section focuses specifically on intrinsic motivation, as it has been used in computational settings. In the Computational Motivation in Swarm and Multi-Agent Settings section, we survey the current approaches into multi-agent and swarm settings and discuss the possible extensions.

Motivation and Intrinsic Motivation From a Psychological Perspective

Ryan and Deci (2000a) succinctly defined motivation "*to be motivated means to be moved to do something.*" Motivation is

TABLE 1 | Summary of reviews on intrinsic motivation in artificial agents and contribution of our survey.

Survey	Comment
Oudeyer and Kaplan, 2007	Provides typology based on motivation theory
Oudeyer et al., 2007	Categorizes existing approaches based on learning and meta-learning
	Highlights difference between knowledge-based vs. competence-based motivation models
Schmidhuber, 2010	Provides typology based on reward theory
Barto, 2013	Discusses intrinsic motivation with relation to reinforcement learning
Merrick, 2017	Intrinsic motivation is surveyed as a part of computational value systems
Roohi et al., 2018	Intrinsic motivation is surveyed as a tool to build player models in computer games
In this study	Introduces computational motivation in a social context

the mechanism that makes humans and animals commit various tasks. Motivation shapes our behavior throughout our life. On the simplest terms, motivation may seem like a straightforward tool that helps an organism to survive. However, a little deliberation can show us the depth, variety and effect of motivation on a grander scale. For example, motivations vary in degree and in type for different people. In a classroom situation, two students doing their homework can be motivated by completely different influences. One of the students may finish the homework for getting high scores in tests, and the other may do it as she finds the subject highly intriguing. Furthermore, a third student may be influence by a combination of both of these motivations, along with a multitude of others.

Researchers from various fields have tried to explain motivation from their respective view. A plethora of concepts on the definition, function, and characteristics of motivation is provided by ethologists (Epstein, 1982), psychologists (Ryan and Deci, 2000a) neuroscientists (Watts and Swanson, 2002; Daw and Shohamy, 2008), and behavioral neuroscientists (Berridge, 2004), among others. The psychological perspective is particularly relevant to our survey. We only provide a brief overview of the psychological theories here. A comprehensive review can be found in Savage (2000).

One of the most influential theories of motivation in psychology is the drive concept, most productively formulated by Hull (1951, 1952). The drive theory states that behavior of an organism is motivated by drives such as hunger and thirst. These drives arise as a response to reduce physiological needs and they motivate behavior which results in the necessary deficit. The theory of drive is centered on the concept of "homeostasis" (Cannon, 1932), where "*bodily conditions are maintained in approximate equilibrium despite external perturbations*." This theory, which says motivated behavior is a response to encounter changes in equilibrium condition, has influenced many other theories on motivation. Moreover, as described by Savage (Savage, 2000), the theory of drive is an attractive option to model motivational systems for artificial agents. The reason for this is the reduction of drive can be translated to a system's reward mechanism which monitors different variables and respond appropriately (i.e., reduce the deficit in the particular need) when they change. However, because drive theory only explains a subset of animal and human behaviors, other theories have been proposed.

Another motivational theory (Toates, 1986), defines motivation as a multiplicative combination of internal state and an incentive factor. According to this, motivation in an organism will arise as an interaction between an internal state of the organism (e.g., thirst) and an external incentive factor (availability of drinking water). Other motivational theories include the hedonic theories, which state we seek pleasurable activities and keep away from the unpleasant ones.

The shortcomings of the drive theory of motivation become evident if we consider certain human and animal behavior. Human infants perform activities that are not driven toward reducing needs such as hunger or thirst. In one experiment (Harlow et al., 1950), a group of monkeys spontaneously attempted to solve a complex puzzle without any specific reward. Likewise, humans perform various activities such as painting, traveling, and playing sports, which do not directly bring any obvious external rewards. This kind of curious and exploratory behavior are not well-modeled by the drive theory of motivation. White (1959) and Berlyne (1960, 1966) pointed out the abundance of such intrinsically rewarding activities that are driven by curiosity, play and exploration and in absence of explicit reward. This is where the notion of intrinsic motivation gets introduced. Intrinsic motivation is defined as the mechanism that encourages organisms to perform an activity "*for the inherent satisfaction rather than some separable consequence*" (Deci and Ryan, 1985). Intrinsically motivated activities are conducted for the fun and challenge rather than achieving external rewards.

As soon as we introduce the notion of intrinsic motivation, the next question is- "what are the factors that make an activity intrinsically motivating?" Psychologists have proposed quite a few theories in this context. Influenced by drive theory, some psychologists proposed these activities are caused by "drive for exploration" (Montgomery, 1954) and "drive to manipulate" (Harlow et al., 1950). However, as criticized by White (1959), these approaches have shortcomings. Indeed, these exploratory activities are not homeostatic, in contrast to what drive theory has proposed. An alternative stream of the idea to explain intrinsically motivated activities is that of "optimal level theories". Dörner and Güss (2013) conducted an experiment that involved rats going through various stimulating activities. The experiment provided some key ideas toward intrinsic motivation. It was observed that if animals continue getting a certain level of environmental complexity, they become used to it and eventually gets bored. If they are provided with a slightly complex stimulus, they become curious again. However, if they encounter a stimulus which is too complex compared to their current situation, it confuses them and they tend to avoid it. In effect, an animal will be intrinsically motivated by the activities and stimuli that are optimally difficult and sit in the middle between familiarity and extreme novelty. Berlyne (1960) explored similar notions. As pointed out by Barto (2013), optimal level theories have important applications in varied areas such child development, architecture, city planning, music, and so on.

Deci and Ryan (1985) further presented the Cognitive Evaluation Theory (CET). CET states that intrinsically motivating activities are the ones that satisfy innate psychological needs such as competence, autonomy and relatedness. In classroom situations, factors such as competence, autonomy and self-determination facilitate intrinsic motivation whereas threats, deadlines, competitions, and tangible rewards diminish intrinsic motivation (Ryan and Deci, 2000a). Berlyne (1966) suggested novelty, incongruity, surprise and complexity as underlying factors that affect intrinsic motivation. As we will see in the latter part of this paper, these factors are extensively used in modeling computational intrinsic motivation.

We have detailed the psychological theories behind intrinsically motivated activities and their distinction from extrinsic motivations. A relevant question after this discussion can be- what are the effectual differences between intrinsic and extrinsic motivation and why would one be interested in intrinsic motivation? One answer to this was provided by Baldassarre (2011). He argued that from an evolutionary perspective, extrinsic motivations guide learning of skill, knowledge and behavior which directly increases the “fitness” (defined as survival and reproductive chances) of an individual, whereas intrinsic motivation produces behavior that increase the fitness only at a later stage. We believe this aspect makes intrinsic motivation more complex and interesting. If we want to build human-like artificial intelligent systems, we need to implement mechanisms inducing intrinsic motives and learn skills and knowledge that may not seem useful now. This is where we need to connect the psychological concepts with the computational models.

Structuring Existing Approaches to Computational Motivation

In this section, we provide a review of the current computational models of motivation from four different aspects: setting, mechanism, function and evaluation. We base our discussion on the concept of an agent which can sense the state of its world, reason about this state and act. While the motivation mechanism, i.e., how motivation is defined, is central to the broader idea, the peripheral concepts are quite relevant as well. **Figure 1** illustrates these concepts, which are further defined in the sections below. The significance of each of these aspects is described in the relevant section, followed by the salient features in the existing work.

Setting

The first aspect we examine is the settings in which computational models of motivation are used. By setting, we mean the artificial intelligence framework into which computational motivation has been embedded. Examples include learning and planning settings. In **Table 2**, we list a cross-section of the areas where computational models of motivation are used. The rows of the table are not mutually exclusive—a single reference can be present in multiple rows.

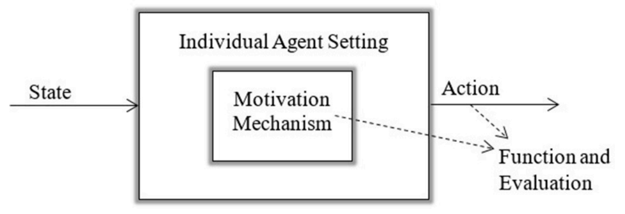


FIGURE 1 | Visualization of the relationship between Setting, Mechanism, Function, and Evaluation. The setting is the artificial intelligence framework wrapped around the motivation mechanism. The motivation mechanism influences the action of the agent, which together produces a measurable function.

TABLE 2 | Settings in which computational models of motivation are used.

Settings	References
Reinforcement Learning	Barto et al., 2004; Simşek and Barto, 2006; Schembri et al., 2007; Sequeira et al., 2011; Kompella et al., 2012; Baldassarre and Mirolli, 2013; Metzen and Kirchner, 2013; Di Nocera et al., 2014; Frank et al., 2015; Hester and Stone, 2015
Deep Learning	Mohamed and Rezende, 2015; Kulkarni et al., 2016; Achiam and Sastry, 2017; Zhelo et al., 2018
Hierarchical Structure	Schembri et al., 2007; Baranes and Oudeyer, 2010; Baldassarre and Mirolli, 2013; Santucci et al., 2013; Frank et al., 2015; Kulkarni et al., 2016
Active Learning	Oudeyer et al., 2007; Baranes and Oudeyer, 2009, 2010, 2013; Kompella et al., 2017; Pathak et al., 2017
Motion Planning	Frank et al., 2015
Affordance Discovery	Hart et al., 2008; Hart, 2009
Goal Discovery/Goal Generation	Salgado et al., 2016; Santucci et al., 2016; Kompella et al., 2017
Multiple skill learning	Santucci et al., 2013
Attention Allocation	Di Nocera et al., 2014; Gatsoulis and Mcginnity, 2015

In Reinforcement Learning (RL), an agent learns from experience as it deals with a sequential decision problem. The agent interacts with an “environment” which contains a “critic” that provides the agent with rewards by evaluating the behavior. Through trial-and-error, the agent maximizes the reward over time. With the introduction of intrinsically motivated reinforcement learning (IMRL; Singh et al., 2010), the reward is designed to be a combination of *extrinsic reward* and *intrinsic reward.* While the extrinsic rewards are closely related to the environment itself, the intrinsic reward is used to introduce the effect of factors that are considered to underlie intrinsic motivation. These include novelty, surprise, incongruity etc., which are relative to the agent’s learning and memory. With this approach, the intrinsic reward is brought to the fore. Though it may not be directly related to the task the agent is supposed to accomplish, these reward can provide the agent with information that are useful to improve performance (Sorg et al., 2010). This philosophy is in line with the psychological perspectives on intrinsic motivation. Deep neural networks have been used to provide "rich representations" for high-dimensional RL tasks successfully (Mnih et al., 2016). IMs mechanisms have been used as reward functions in Deep Reinforcement Learning (DRL). Intrinsic Motivation in DRL showed improved performance in challenging environments with sparse rewards by providing efficient exploration strategies.

Many of the computational models of IM have organized their structures hierarchically. Typically, a two-level or two-module structure is used. This approach makes it possible to compartmentalize IM from sensorimotor learning. For example, the upper level module generates IM based goals while the lower level explores the environment for reaching particular goal and necessary actions.

In many real-word scenarios, reinforcement learning agents suffer due to the extremely sparse nature of extrinsic rewards. Besides, in an open-ended learning scenario where we would want an agent to learn from the large sensorimotor space by itself. This calls for active learning approaches which can guide an agent by organized and constrained exploration. Intrinsic motivation is used in active learning as a heuristics that helps to maximize learning progress.

Real-world scenarios pose similar problems to artificial agents in many of the other fronts as well. These agents need to be able to plan their motion by finding a path among arbitrary and previously unknown obstacles. Exhaustive searching is computationally expensive- this makes the agents slow, which is quite the opposite of the features we would like to see in a humanoid robot.

Another related trait is an organism's ability to perceive its environment and interact with it. In real world, this would translate to a robot's ability to navigate and using tools in appropriate manner by itself. This resulted in to apply intrinsic motivation to discover object utilities and use them accordingly.

A hallmark of human intelligence is the ability to self-determine goals to achieve particular skills. To be truly autonomous, an artificial system has to discover and select goals on its own. Based on the complexity of these goals, the system may need to decompose it into sub-goals. These sub-goals may not have any tangible rewards at a certain time. Thus, goal discovery and identification from a large space and working toward that by identifying sub-goals become a complex proposition. Intrinsic motivation can be useful in this regard.

While skill acquisition for artificial agents is a complex issue itself, it gets more complicated with the presence of multiple learnable skills. While for humans it is intuitive to choose to learn skills in terms of increasing need and complexity, this is difficult for artificial agents. Closely related to this is the ability to focus attention on a task that is learnable and useful. Computational models of motivation can be used in these cases as well.

The settings listed in **Table 2** denote fields which carry significant importance in autonomous mental development. A true autonomous human-like autonomous agent must be able to plan its motions in the continuous space while taking into consideration its own constraints. Throughout its lifetime, it will encounter novel objects and will have to learn about the affordances of objects and how to manipulate them. To acquire competence, it needs to be able to identify goals by itself, learn to compose multiple skills into more complex ones, and allocate attention to learnable situations. As we have predicted, this table provides us with aspects that are fundamental to design machines that help an artificial agent appear to think and act like a human.

TABLE 3 | Reward mechanisms in computational models of motivation.

Mechanism	Reference
Prediction error	Barto et al., 2004; Metzen and Kirchner, 2013
Empowerment	Salge et al., 2014; Mohamed and Rezende, 2015
Learning Progress/Information Gain/KL Divergence	Oudeyer et al., 2007; Baranes and Oudeyer, 2009; Kompella et al., 2012; Frank et al., 2015; Hester and Stone, 2015
Curiosity	Kompella et al., 2012; Di Nocera et al., 2014; Frank et al., 2015; Pathak et al., 2017; Zhelo et al., 2018
Novelty	Metzen and Kirchner, 2013; Gatsoulis and Mcginnity, 2015; Hester and Stone, 2015; Salgado et al., 2016
Surprise	Schembri et al., 2007; Hamann, 2015; Achiam and Sastry, 2017

Mechanism

As we said earlier, in this paper we view the motivation mechanism as a goal generator. It takes the agent's sensations of its world state, and memories of past sensations, as inputs and generates goals that in turn influence action selection. These goals have been expressed in different ways in the literature, ranging from explicit goal structures to implicit utility feedback.

In this section, we describe a number of computational models of intrinsic motivation with respect to the specific nature of the motivation mechanism. A list of possible formal mechanisms of intrinsic motivations was provided in Oudeyer and Kaplan (2007). The list in **Table 3** mostly concurs with that typology. The intrinsic reward mechanism works as a part of the organism itself in the reinforcement learning framework. Rewards based on novelty, curiosity and uncertainty are defined with respect to the visited states. Hester and Stone (2015) measured novelty as the distance of the unexplored region of states from the previously visited states. The intrinsic reward is given as proportional to this distance. This motivates the agent to explore the state-actions that are the most different from the ones that are already visited.

Using prediction error is inspired by dopamine neurons. In its most basic form, the reward for an event is proportional to the error that was made for a certain event. As agents make prediction of future events based on current ones, this intrinsic reward can enable an agent to focus on an event that has a larger error associated with it. As an agent repeatedly learns more about the event and achieves more success, the intrinsic reward decreases.

Empowerment is a measure of the causal influence an agent has on the perceived world (Klyubin et al., 2005). In computational motivational models, empowerment is typically implemented as a measure of maximizing information or minimizing uncertainty. This provides the agents with adaptability to deal with dynamic environments.

Learning progress or information gain is another highly used reward measure. In this case, the system generates rewards if predictions improve over time, i.e., it tries to minimize prediction errors. By doing this, an agent can focus on states or activities that offer the highest progress. Using learning progress as a reward measure is a robust solution to changing environments. Moreover, learning progress based mechanism result in strategies that autonomously progress from simpler to more complex tasks (Gottlieb et al., 2016).

Curiosity is one of the most widely used measurements of computational intrinsic motivation. In effect, curiosity is defined as a function of learning progress or prediction error. In curiosity-driven exploration, agents are intrinsically rewarded to explore regions which shows higher learnability. Curiosity-driven agents are demonstrated to learn even in situations without any extrinsic rewards (Pathak et al., 2017).

Novelty can be an effective mechanism to implement intrinsic motivation. By encouraging agents to explore states that are highly different than the already visited ones, efficient exploration can be achieved. Measuring novelty typically involves a comparison between the current stimuli and the previous ones. Furthermore, it can also include factors such as habituation, which involves the temporal effects of similar stimuli on novelty.

Surprise is defined as the difference between expectation and outcome. In that case, prediction error can be used as a measurement of surprise and intrinsic rewards. In some other models, surprise is defined as the degree of not expecting an incident.

Some of the implementations have combined the aforementioned reward measures and defined their own (Baranes and Oudeyer, 2010; Sequeira et al., 2011; Hester and Stone, 2015).

Function

In this category, we provide a list of functions that result from the implementation of a motivation mechanism in artificial agents. We opted for this aspect with the idea that it would complement the settings that we described previously.

One of the major functions imparted by intrinsic motivation is that of efficient exploration. Agents demonstrate features such as achieving significant states with sparse and delayed rewards, scalability in computationally extensive scenarios. In case of autonomous self-organization, agents could discover their sensorimotor skills by virtue of intrinsic motives without any explicit guidance. It was also shown that computational models of intrinsic motivations foster progressive learning. Intrinsically motivated agents initially spend time in easier situations and then allocate attention to situations with increasing difficulty. This tendency is directly related to the next function of composing complex task by learning simpler tasks first. Artificial agents with intrinsic motivations were significantly faster in completing complex tasks.

The features are listed in **Table 4**. Various autonomous activities feature prevalently here, with exploratory actions topping the list. This concurs with the very nature and definition of motivated behavior. Intrinsic motivation is supposed to foster exploration and curious behavior that may or may not aid immediate competence and skill acquisition. This demonstrates that existing implementations of computational motivation, irrespective of the settings or reward mechanism, introduces exploratory actions in artificial agents.

TABLE 4 | List of functions resulted from computational models of intrinsic motivation.

Function	References
Efficient Exploration	Frank et al., 2015; Hester and Stone, 2015; Mohamed and Rezende, 2015; Kulkarni et al., 2016; Salgado et al., 2016; Santucci et al., 2016; Achiam and Sastry, 2017; Pathak et al., 2017
Autonomous self-organization	Oudeyer et al., 2007
Progressive learning	Oudeyer et al., 2007; Hester and Stone, 2015
Task composition	Schembri et al., 2007

TABLE 5 | Methods used to evaluate computational models of intrinsic motivation.

Evaluation methods	References
Comparison with extrinsic reward	Cameron and Pierce, 1994; Barto et al., 2004; Di Nocera et al., 2014; Hester and Stone, 2015
Goal accomplishment	Kulkarni et al., 2016
Comparison with random and least tried states	Frank et al., 2015
Comparison with greedy approach	Sequeira et al., 2011
Comparing single and multiple intrinsic rewards	Sequeira et al., 2011
Analyzing performance over time	Gatsoulis and Mcginnity, 2015

Evaluation Methods

The behavior of an agent comprises the sequences of actions it performs. Evaluating the behavior of an agent driven by computational motivation is not a straightforward task. In case of a motivated agent, we not only want to measure the completion rate for a particular task, but also intend to observe, and measure, the effect of the intrinsic reward on the agent's behavior, organization, and long-term competence and knowledge acquisition. The evaluation methods used in the existing literature are listed in **Table 5**.

As a baseline, many of the proposals compare the intrinsically motivated behavior with that of extrinsic reward only. While this largely demonstrates the effectiveness of intrinsic rewards, one may fail to understand the particular influences of intrinsic motivation if it is not extended further. Similar evaluation approaches include comparing intrinsically motivated behavior with random or greedy method. Novel tools to aid the research in computational intrinsic motivation are proposed by groups of researchers from various fields (Natale et al., 2013; Stafford et al., 2013; Taffoni et al., 2013).

Computational Motivation in Swarm and Multi-agent Settings

While the previous sections examined literature on computational models of motivation, here we focus on the concept of motivation in a social context. This can be a swarm of robots, particles in an optimisation setting, or a multi-agent setting. By providing a summary of the existing works, we point out the fact that in these cases, computational motivation has been used in a limited capacity, almost identical to that of single agent scenarios. We then discuss few of the possible extensions of computational motivation that can be applicable to swarm and multi-agent settings.

Merrick (2015) demonstrated the effects of motive profiles in game-playing agents. With the help of two-player game settings, it was shown that power, achievement, and affiliation motives can lead to various emergent behaviors. Moreover, in case of evolutionary algorithm for creating motivated agents, it was shown that motivated agents were more diverse and achieve higher incentive than their non-motivated versions. Evolutionary models of intrinsically motivated agents were simulated in a multi-agent setting in by Shafi et al. (2012). A static incentive model was used to generate agents that change their motives over time.

Hardhienata et al. (2012, 2014a,b) incorporate achievement, power, affiliation and leadership motivation in a Particle Swarm Optimization (PSO) setting. The application area the agents are tested in is task allocation. Incentives are defined as distance of the agent from the task and the number of agents around the task. The incentive value, along with motive profiles, help inform an agent which task and neighborhood to choose. Their work shows the effect of motivational profiles on an established algorithm such as PSO and task allocation.

Klyne and Merrick (2016) used computational models of motivation to generate dynamic fitness functions for PSO. The convergence of the swarm on the generated fitness function is tested in a workplace hazard mitigation scenario. The two approaches to model motivation in this framework were novelty and curiosity. Curiosity was implemented via K-Means neural network. The results showed this approach is more applicable for decentralized data sources. In case of a centralized source, background subtraction was used. This is an image processing technique to detect novel objects in successive image frames. Evaluation metrics include ability of goal generation and convergence of swarm once the goal is generated. In some cases, the generated fitness function has too many local maxima for the swarm to converge on the right place.

Saunders and Gero (2004) used flocking and a social force model to design curious agents. These agents perform evaluations of environments which are designed to prompt exploration, e.g., art galleries. Interestingness is modeled using the Wundt curve: the most interesting situations are the ones that have moderate novelty. Situations are measured with hedonistic values and agents will move toward stimuli having higher value. Their results showed that these curious agents spend significantly more time in environments which were designed "better." Linkola et al. (2016) used novelty to create a group of creative and curious agents. A society of homogenous agents is created, with each one having an individual memory. The behavior of the population is modeled via iterations. In each iteration, each agent creates a candidate artifact bases on the current location and memory. The agents then collectively decide which of the candidate artifacts can be added the repository. The agents are self-critical and have veto power. As the results show, self-criticism lowers the amount of collaborative effort in evaluating candidate artifacts while veto power increases novelty. Galvao et al. (2015) implement the notion of novelty search (Lehman and Stanley, 2011) in PSO.

Table 6 summarizes the existing work that incorporates motivation in multi-agent and swarm settings using the headings of setting, mechanism, and function introduced in the previous section. As the settings column demonstrates, most of the works have been implemented in PSO and game-theoretic settings. Moreover, in many of these works the focus was on the function rather than the motivation mechanism. As a result, they lack a detailed analysis of the implications of motivation being implemented in a social context. This is where our proposals of the newer settings and possibilities come in.

Figure 2 hypothesizes about the new functions that we may achieve in motivated agents if augment the field with new multi-agent and swarm settings. The two extremes of the vertical axis in **Figure 2** represent the knowledge-based and competence-based mechanisms of intrinsic motivation. As we pointed out earlier, this is the most general categorization of the computational models, which covers a range of detailed mechanisms discussed in Section Mechanism. In the upper part of the vertical axis, we have the knowledge-based models which produce exploratory functions and acquire knowledge about the surrounding environment or the world. In the lower part, we have the competence-based models which function to improve skills. On the other hand, the horizontal axis represents an expanded view of possible e agent settings. On the right-hand side, we have individual agent settings. On the left side, we introduce multi-agent and swarm settings. We hypothesize that these settings will see motivated agents performing new functions including leadership, for example leading agents that do not have intrinsic motives, and scaling and communication functions. By this we mean that extension to the multi-agent settings permits intrinsically motivated agents to scale to problems that cannot be solved by a single agent.

In addition to enabling new functions, we hypothesize a range of ways that motivation may be embedded in multi agent or swarm settings. For example, motivation may be distributed among multiple agents or it may be shared. In the first case, the motivation mechanism of each agent is processed and acted upon by itself. The group behavior rules, such as flocking, will still affect the emergent behavior. In case of shared motivation, agents will interact with each other while constructing the motivated behavior. The underlying assumption in this case is that the interpersonal factors, such as aligning one's goal to a friend's interests, will play an important factor to determine the group behavior. With the introduction of

TABLE 6 | A summary of the work using motivation in social context in terms of their setting, function and mechanism.

Setting	Function	Mechanism	Reference
Various Games	Game theoretic analysis	Motive Profiles	Merrick, 2015
Prisoner's Dilemma	Game theoretic analysis	Motive Profiles	Shafi et al., 2012
PSO	Task Allocation	Motive Profiles	Hardhienata et al., 2012, 2014a,b
PSO	Generating dynamic fitness function	Curiosity Novelty	Klyne and Merrick, 2016
Flocking, Social Force Model	Design Evaluations	Curiosity	Saunders and Gero, 2004
Multi-agent systems	Generating Creative Artifacts	Novelty	Linkola et al., 2016
PSO	Grammatical Swarm	Novelty	Galvao et al., 2015

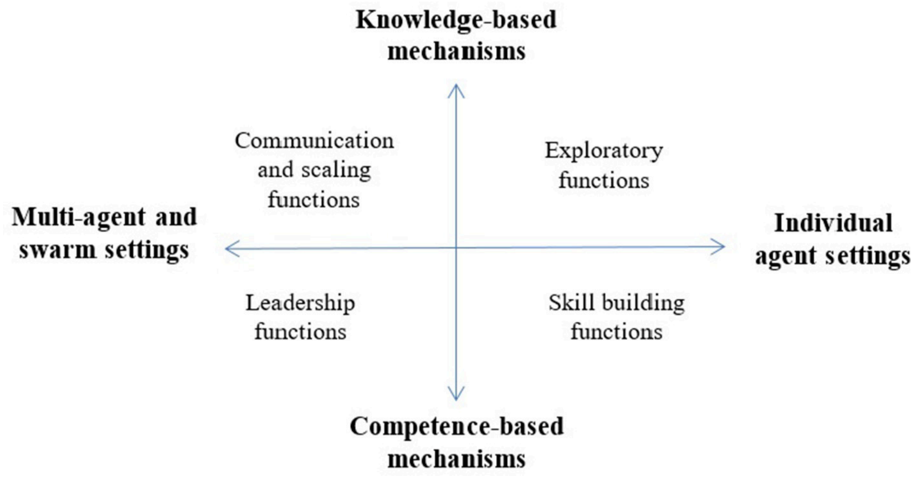

Knowledge-based mechanisms
Communication and scaling functions
Exploratory functions
Multi-agent and swarm settings
Individual agent settings
Leadership functions
Skill building functions
Competence-based mechanisms

FIGURE 2 | Expanding the view of settings for computational models of motivation also expands possible functions.

this concept, we can explore notions such as conformity, divergence and living up to the expectations of others. These concepts have been investigated in human psychology (Fishbach et al., 2016) but not yet in the computational motivation research.

Moreover, all agents in a group may have the same motives or they may have different motives. While some agents in a group can be motivated to improve personal knowledge and competence, others can pursue that knowledge for gaining control over the group. This can result in homogeneous and heterogeneous groups of motivated agents. Some of the existing works investigated the effects of different motives but only a few looked into heterogeneously motivated artificial agents coexisting as a group. Likewise, all agents may be motivated agents or only a subset of agents may be motivated, and others may not possess such models. These variants are illustrated in **Figure 3**.

DISCUSSION

In this section, we discuss the challenges associated with computational modeling of intrinsic motivation in social context and conclude the paper.

Research Challenges

There are quite a few challenges that emerge as we ponder the notion of multiple or swarms of agents equipped with computational motivation. We discuss these challenges from the four aspects we had described in section Structuring Existing Approaches to Computational Motivation- settings, rewards, functions, and evaluation. Combining these aspects with the possible extensions discussed in section Computational Motivation in Swarm and Multi-Agent Settings, we present a set of research challenges. These research challenges encapsulates our discussion in the previous sections and provide an overview of the future research involving computational motivation in a social context.

Settings to Accommodate the Social Context

Most of the existing works on computational motivation have used PSO or flocking as setting. Though they provide a structured base to investigate motivation, they are too limited in many ways. Human and animal motivation mechanisms involves interaction patterns that are significantly more complex than these restricted settings. We feel eventually there will be a need to generate more flexible, complex, and accommodating multi-agent settings. Settings such as Belief-Desire-Intention (BDI) architecture (Rao and Georgeff, 1991), game theory (Parsons

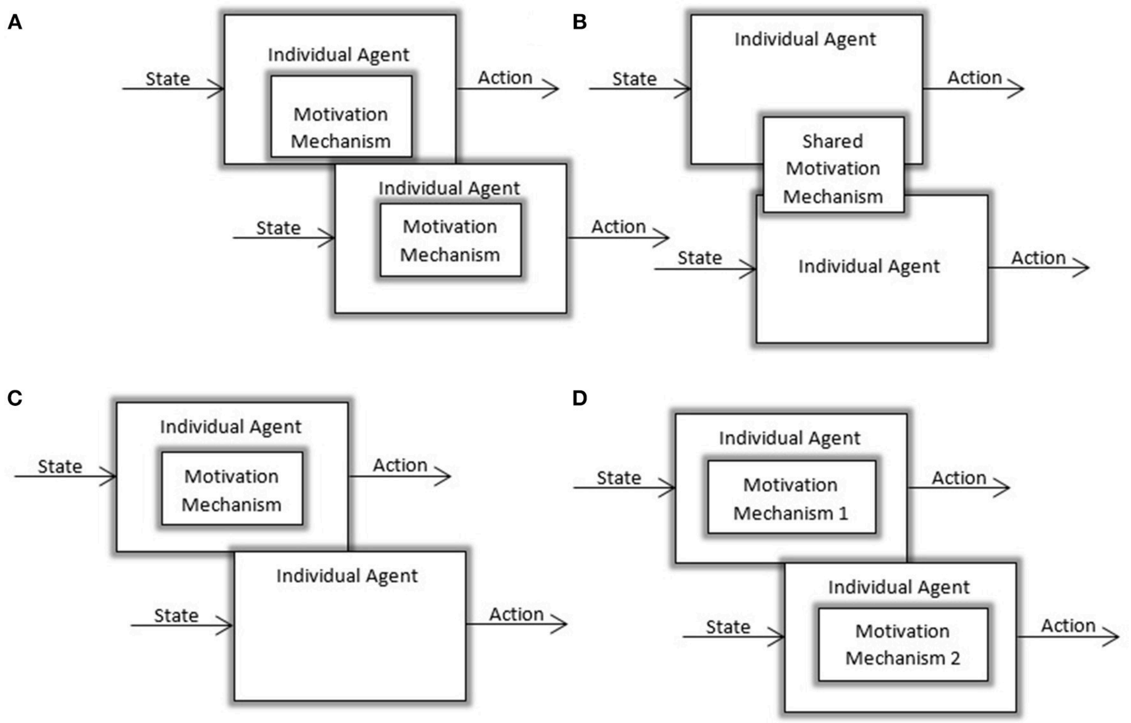


FIGURE 3 | Visualization of different possible multi-agent settings: **(A)** Multi-agent setting with homogeneous agents and decentralized motivation mechanisms **(B)** Multi-agent settings with homogeneous agents using a centralized motivation mechanism **(C)** Heterogeneous society with a subset of motivated agents **(D)** Heterogeneous society with agents using different motivation mechanisms.

and Wooldridge, 2002), multi-agent reinforcement learning (Busoniu et al., 2008), and swarm intelligence (Brambilla et al., 2013) can be useful in this context. Researchers from these respective areas will have to identify the existing properties and extend the architecture to accommodate computational models of motivation.

As the notion of multiple motivated agents is introduced, quite a few features for these settings can be proposed. These were not applicable to single agents, but pertinent in multi-agent scenarios. As discussed previously, we might have a homogenous set of agents equipped with the same motivation mechanism. Many of the current works are exploring this line of work where each agent or particle in a swarm has the same motivation. However, there can be a heterogeneous setting where the agents can be motivated through varying mechanisms of motivation. For example, one agent can act as an informed individual and the other members of the group can be following that agent for achieving a goal or skill. Though there are current works that propose and measure the effectiveness of various motive profiles, there is no study that focuses on a swarm of agents that are equipped with different forms of computational motivation. Another related open area of research will consist of scenarios that will be able to accommodate a temporal change of the motivation mechanism. Imagine a scenario where agents start with having affiliation motive as the primary driving factor but changes to power motives after gaining certain knowledge or skillset.

Novel Mechanisms and Rewards

The current work on the computational models of motivation is largely comprised of translating the psychological theories of motivation into mathematical and computational models. Motivation can have many facets and dimensions when it is compounded by social factors such as presence of and interaction with other individuals. There are some works on various motive profiles, as we discussed in the earlier sections. However, the social theories of motivation are yet to be generally implemented as computational models.

Psychological studies in organizational behavior, student motivation and performance reveal interesting facts about the mechanism of intrinsic motivation in social contexts. It has been observed that support for competence, relatedness and autonomy helps increase children's intrinsic motivation (Ryan and Deci, 2000b). In an educational environment involving high-school students, the authors showed that optimal challenge and performance feedback facilitates competence while relatedness is increased by meaningful parental involvement and peer acceptance (Dörner and Güss, 2013). If we want to model motivation in a social context, we would need to utilize the psychological studies and introduce novel reward mechanisms such as relatedness, feedback and autonomy support.

The next generation of computational models would need to implement, and in many ways, extend and augment the psychological theories. Motivated artificial agents can provide both psychologists and computational intelligence researchers with an avenue of proposing and evaluating various psychological theories. A swarm of motivated agents in a simulated environment can be used to model and predict motivational tendencies such as curiosity and novelty. This can be an alternative to questionnaire-based analysis that typically takes place in psychological studies.

Functions

The existing works have shown the inception of various emergent functional behavior as a result of computational motivation mechanism. With multiple agents, the emergent functions are more versatile, complex and significant. This is due to the fact that within a social context the agents are interacting not only with the environment but also within themselves. Factors such as shared goals and conflicting motivations can lead to interesting emergent behaviors in swarms and multi-agent systems. Seemingly simple and primitive rules can be combined to produce complex behavior patterns. One challenge of the future research would be to devise mechanism to generate such behaviors from the primitive rules and through motivation mechanisms. For example, one can think of a flock of agents swarming through an environment by virtue of the intrinsic motivation values. In this case, we will need to design, estimate, and adapt the effects of motivational mechanisms on individual agent as well as on the group behavior. While it would have been relatively simpler to achieve this in an individual agent setting, it would be much more complex in case of multiple interacting agents. It will be interesting and useful to observe the effect of different motivation mechanisms on this mapping between primitive rules and emergent behavior.

Evaluating Behaviors

As we have pointed out already, evaluating the consequences of implementing motivation (especially intrinsic motivation) is not quite straightforward. The state-of-the-art single agent architectures still suffer from a lack of widely accepted behavior metrics capable of measuring the effects of computational motivation. With multiple motivated agents, the challenge of measuring emergent motivated behavior becomes increasingly complex. It would be a challenge to determine behavior metrics that can embody the motivated behavior of multiple interacting agents.

We emphasize on the evaluation as it has significant implications involving the motivation mechanism and expected behavior. A causal relation between the motivation mechanism and behavior can be used to generate artificial agents with intended motive profiles and tune them as need be. The current proposals do not establish a rigorous or mathematical relation between the motivation parameters (e.g., reward measures) and the corresponding behavior (e.g., skills acquired). As more complex scenarios involving multiple agents are introduced, this becomes even more challenging. If the next generation of these proposals can determine the relation between the controlling parameters and emergent behavior more rigorously, we will be able to derive more applications of the motivated agents. Determining appropriate behavior metrics will play a significant part in this regard.

CONCLUSION

Motivated artificial agents are designed to acquire knowledge and skills in an open-ended setting. These features can provide new horizons in artificial intelligence, machine learning and computational intelligence in general. In this survey, we have summarized the implementations of motivation in artificial agents. We provided definitions, background, and state-of-the-art of the field of computational motivation. We have provided a new typology through which the current research can be categorized through four main aspects: setting, mechanism, function, and evaluation method. Through a systematic discussion, we demonstrated the fact that there is limited work using computational motivation in multi-agent and swarm settings. Following a detailed discussion on this topic, we presented the major research challenges for achieving societies of multiple motivated agents. We believe our contribution in this paper will help researchers to further identify and explore these open research topics.

AUTHOR CONTRIBUTIONS

MK and KK conceived the idea of the paper. MK conducted the analysis and wrote the first draft. KK and MB provided critical feedback and suggestions to organize the subsequent revisions. KK contributed several diagrams and related texts. KK and MB overviewed the whole paper writing process. MK finalized the paper.

REFERENCES

Achiam, J., and Sastry, S. (2017). *Surprise-Based Intrinsic Motivation for Deep Reinforcement Learning. arXiv e-prints [Online]*. Available online at: https://ui.adsabs.harvard.edu/#abs/2017arXiv170301732A (Accessed March 01, 2017).

Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., et al. (2009). Cognitive developmental robotics: a survey. *IEEE Transact. Autonom. Mental Dev.* 1, 12–34. doi: 10.1109/TAMD.2009.2021702

Baldassarre, G. (2011). "What are intrinsic motivations? a biological perspective," in *IEEE International Conference On Development And Learning* (Icdl), IEEE. 1–8. doi: 10.1109/DEVLRN.2011.6037367

Baldassarre, G., and Mirolli, M. (2013). *Deciding which skill to learn when: temporal-difference competence-based intrinsic motivation (Td-Cb-Im). Intrinsically motivated learning in natural and artificial systems.* Berlin; Heidelberg: Springer, 257–278. doi: 10.1007/978-3-642-32375-1_11

Baranes, A., and Oudeyer, P.-Y. (2010). "Intrinsically motivated goal exploration for active motor learning in robots: a case study," in *IEEE/Rsj International Conference On Intelligent Robots And Systems* (Taipei: IEEE), 1766–1773.

Baranes, A., and Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robot. Autonom. Syst.* 61, 49–73. doi: 10.1016/j.robot.2012.05.008

Baranes, A., and Oudeyer, P. Y. (2009). R-Iac: robust intrinsically motivated exploration and active learning. *IEEE Trans. Autonom. Mental Dev.* 1, 155–169. doi: 10.1109/TAMD.2009.2037513

Barto, A. G. (2013). *Intrinsic Motivation And Reinforcement Learning. Intrinsically Motivated Learning In Natural And Artificial Systems.* Berlin: Springer, 17–47. doi: 10.1007/978-3-642-32375-1_2

Barto, A. G., Singh, S., and Chentanez, N. (2004). "Intrinsically Motivated Learning Of Hierarchical Collections Of Skills," in *Proceedings Of The 3rd International Conference On Development And Learning* (San Diego, CA), 112–119.

Beekman, M., Sword, G. A., and Simpson, S. J. (2008). "Biological foundations of swarm intelligence," in *Swarm Intelligence: Introduction and Applications* ed D. Merkle (Heidelberg: Springer), 3–41.

Beni, G. (2004). "From swarm intelligence to swarm robotics," in *International Workshop On Swarm Robotics* (Santa Monica, CA: Springer), 1–9.

Beni, G., and Wang, J. (1993). *Swarm intelligence in cellular robotic systems. robots and biological systems: towards a new bionics?* Tuscany: Springer, 703–712. doi: 10.1007/978-3-642-58069-7_38

Berlyne, D. E. (1960). *Conflict, Arousal, and Curiosity*. Mcgraw-Hill Series in Psychology. New York, NY: Mcgraw-Hill Book Company.

Berlyne, D. E. (1966). Curiosity and exploration. *Science* 153, 25–33. doi: 10.1126/science.153.3731.25

Berridge, K. C. (2004). Motivation concepts in behavioral neuroscience. *Physiol. Behav.* 81, 179–209. doi: 10.1016/j.physbeh.2004.02.004

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intellig.* 7, 1–41. doi: 10.1007/s11721-012-0075-2

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man Cybernet. Part C Appl. Rev.* 38, 156–172. doi: 10.1109/TSMCC.2007.913919

Cameron, J., and Pierce, W. D. (1994). Reinforcement, reward, and intrinsic motivation: a meta-analysis. *Rev. Educ. Res.* 64, 363–423. doi: 10.3102/00346543064003363

Cannon, W. B. (1932). *Homeostasis*. Norton; New York, NY: The Wisdom of the Body.

Dautenhahn, K. (1995). Getting to know each other—artificial social intelligence for autonomous robots. *Rob. Auton. Syst.* 16, 333–356. doi: 10.1016/0921-8890(95)00054-2

Daw, N. D., and Shohamy, D. (2008). The cognitive neuroscience of motivation and learning. *Soc. Cogn.* 26, 593–620. doi: 10.1521/soco.2008.26.5.593

Deci, E., and Ryan, R. (1985). *Intrinsic Motivation and Self-Determination in Human Behaviour.* New York, NY: Springer Science & Business Media.

Di Nocera, D., Finzi, A., Rossi, S., and Staffa, M. (2014). The role of intrinsic motivations in attention allocation and shifting. *Front. Psychol.* 5:273. doi: 10.3389/fpsyg.2014.00273

Dörner, D., and Güss, C. D. (2013). Psi: a computational architecture of cognition, motivation, and emotion. *Rev. Gen. Psychol.* 17, 297–317. doi: 10.1037/a0032947

Epstein, A. N. (1982). "Instinct and motivation as explanations for complex behavior," in *The Physiological Mechanisms Of Motivation,* ed D. W. Pfaff (New York, NY: Springer), 25–58.

Fishbach, A., Steinmetz, J., and Tu, Y. (2016). Motivation in a social context: coordinating personal and shared goal pursuits with others. *Adv. Motivat. Sci.* 3, 35–79. doi: 10.1016/bs.adms.2015.12.005

Frank, M., Leitner, J., Stollenga, M., Förster, A., and Schmidhuber, J. (2015). Curiosity driven reinforcement learning for motion planning on humanoids. *Front. Neurorobot.* 7:25. doi: 10.3389/fnbot.2013.00025

Galvao, D. F., Lehman, J., and Urbano, P. (2015). "Novelty-driven particle swarm optimization," in *International Conference On Artificial Evolution (Evolution Artificielle)* (Lyon: Springer International Publishing), 177–190.

Gatsoulis, Y., and Mcginnity, T. M. (2015). Intrinsically motivated learning systems based on biologically-inspired novelty detection. *Rob. Auton. Syst.* 68, 12–20. doi: 10.1016/j.robot.2015.02.006

Gottlieb, J., Lopes, M., and Oudeyer, P.-Y. (2016). "Motivated cognition: neural and computational mechanisms of curiosity, attention, and intrinsic motivation," in *Recent Developments In Neuroscience Research On Human Motivation,* eds K. Sung-Il, R. Johnmarshall, and B. Mimi (Emerald Group Publishing Limited), 149–172.

Hamann, H. (2015). "Evolution of collective behaviors by minimizing surprisal and by micro-macro links," in *Proceedings Of The Companion Publication Of The 2015 Annual Conference On Genetic And Evolutionary Computation Acm* (Madrid), 1253–1253.

Hardhienata, M. K., Merrick, K. E., and Ugrinovskii, V. (2012). "Task allocation in multi-agent systems using models of motivation and leadership," in *Evolutionary Computation (Cec), 2012 IEEE Congress On, IEEE* (Brisbane, QLD), 1–8.

Hardhienata, M. K., Ugrinovskii, V., and Merrick, K. E. (2014a). "Task allocation under communication constraints using motivated particle swarm optimization," in *Evolutionary Computation (Cec), 2014 IEEE Congress On IEEE* (Beijing), 3135–3142.

Hardhienata, M. K. D., Merrick, K. E., and Ugrinovskii, V. (2014b). "Effective motive profiles and swarm compositions for motivated particle swarm optimisation applied to task allocation," in *2014 IEEE Symposium On Computational Intelligence For Human-Like Intelligence, Cihli* (Orlando, FL), 1–8.

Harlow, H. F., Harlow, M. K., and Meyer, D. R. (1950). Learning motivated by a manipulation drive. *J. Exp. Psychol.* 40, 228–234. doi: 10.1037/h0056906

Hart, S. (2009). "An intrinsic reward for affordance exploration, development and learning, Icdl 2009," in *IEEE 8th International Conference On, IEEE* (Shanghai), 1–6.

Hart, S., Sen, S., and Grupen, R. (2008). "Generalization and transfer in robot control," in *Epigenetic Robotics Annual Conference* (Brighton, UK).

Herrmann, J. M., Pawelzik, K., and Geisel, T. (2000). Learning predictive representations. *Neurocomputing* 32, 785–791. doi: 10.1016/S0925-2312(00)00245-9

Hester, T., and Stone, P. (2015). Intrinsically motivated model learning for developing curious robots. *Artif. Intell.* 247, 170–186. doi: 10.1016/j.artint.2015.05.002

Huang, X., and Weng, J. (2002). *Novelty And Reinforcement Learning in the Value System of Developmental Robots*. Lund: Lund University Cognitive Studies, 47–55.

Hull, C. L. (1951). *Essentials of Behavior*. New Haven, CT: Yale University Press.

Hull, C. L. (1952). *A Behavior System; An Introduction To Behavior Theory Concerning The Individual Organism.* New Haven, CT: Yale University Press.

Kaplan, F., and Oudeyer, P.-Y. (2003). *Motivational Principles for Visual Know-How Development*. Lund: Lund University Cognitive Studies, 73–80.

Klyne, A., and Merrick, K. (2016). Intrinsically motivated particle swarm optimisation applied to task allocation for workplace hazard detection. *Adapt. Behav.* 24, 219–236. doi: 10.1177/1059712316651686

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). "Empowerment: a universal agent-centric measure of control," in *IEEE Congress On Evolutionary Computation,* (IEEE: Edinburgh) 128–135.

Kompella, V. R., Luciw, M., Stollenga, M., Pape, L., and Schmidhuber, J. (2012). "Autonomous learning of abstractions using curiosity-driven modular incremental slow feature analysis," in *Development and Learning and Epigenetic Robotics (Icdl), 2012 IEEE International Conference On, IEEE* (Sand Diego, CA), 1–8.

Kompella, V. R., Stollenga, M., Luciw, M., and Schmidhuber, J. (2017). Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. *Artif. Intell.* 247, 313–335. doi: 10.1016/j.artint.2015.02.001

Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., and Tenenbaum, J. B. (2016). "Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation," in *Proceedings of the 30th International Conference on Neural Information Processing Systems* (Barcelona).

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behav. Brain Sci.* 40:e253. doi: 10.1017/S0140525X16001837

Lehman, J., and Stanley, K. O. (2011). abandoning objectives: evolution through the search for novelty alone. *Evol. Comput.* 19, 189–223. doi: 10.1162/EVCO_a_00025

Linkola, S., Takala, T., and Toivonen, H. (2016). "Novelty-seeking multi-agent systems," in *Seventh International Conference On Computational Creativity* (Paris), 1–8.

Marshall, J., Blank, D., and Meeden, L. (2004). "An emergent framework for self-motivation in developmental robotics," *in 3rd International Conference On Development And Learning (Icdl 2004)*, San Diego, CA, Salk Institute.

Merrick, K. (2015). The role of implicit motives in strategic decision-making: computational models of motivated learning and the evolution of motivated agents. *Games* 6, 604–636. doi: 10.3390/g6040604

Merrick, K. (2017). Value systems for developmental cognitive robotics: a survey. *Cogn. Syst. Res.* 41, 38–55. doi: 10.1016/j.cogsys.2016.08.001

Merrick, K. E. (2013). "Novelty and beyond: towards combined motivation models and integrated learning architectures," in *Intrinsically Motivated Learning in Natural and Artificial Systems,* eds G. Baldassarre and M. Mirolli (Berlin: Springer), 209–233. doi: 10.1007/978-3-642-32375-1_9

Merrick, K. E., and Shafi, K. (2011). Achievement, affiliation, and power: motive profiles for artificial agents. *Adapt. Behav.* 19, 40–62. doi: 10.1177/1059712310395953

Metzen, J. H., and Kirchner, F. (2013). Incremental learning of skill collections based on intrinsic motivation. *Front. Neurorobot.* 7:11. doi: 10.3389/fnbot.2013.00011

Mirolli, M., and Baldassarre, G. (2013a). "Functions and mechanisms of intrinsic motivations," in *Intrinsically Motivated Learning in Natural and Artificial Systems*, eds G. Baldassarre and M. Mirolli (Berlin: Springer), 49–72.

Mirolli, M., and Baldassarre, G. (2013b). "Functions and mechanisms of intrinsic motivations: the knowledge versus competence distinction," in *Intrinsically Motivated Learning In Natural And Artificial Systems,* eds, G. Baldassarre and M. Mirolli, (Berlin: Springer-Verlag), 49–72.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). "Asynchronous methods for deep reinforcement learning," in *International Conference On Machine Learning* (New York, NY), 1928–1937.

Mohamed, S., and Rezende, D. J. (2015). Variational information maximisation for intrinsically motivated reinforcement learning. *Adv. Neural Inf. Process. Syst.* 2, 2125–2133. Available online at: https://dl.acm.org/citation.cfm?id=2969477

Montgomery, K. C. (1954). The role of the exploratory drive in learning. *J. Comp. Physiol. Psychol.* 47, 60–64. doi: 10.1037/h0054833

Natale, L., Nori, F., Metta, G., Fumagalli, M., Ivaldi, S., Pattacini, U., et al. (2013). "The icub platform: a tool for studying intrinsically motivated learning," in *Intrinsically Motivated Learning In Natural And Artificial Systems,* eds G. Baldassarre and M. Mirolli (Berlin: Springer), 433–458. doi: 10.1007/978-3-642-32375-1_17

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evolut. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Oudeyer, P. Y., and Kaplan, F. (2007). What is intrinsic motivation? a typology of computational approaches. *Front. Neurorobot.* 1:6. doi: 10.3389/neuro.12.006.2007

Panait, L., and Luke, S. (2005). Cooperative multi-agent learning: the state of the art. *Auton. Agent Multi. Agent Syst.* 11, 387–434. doi: 10.1007/s10458-005-2631-2

Parsons, S., and Wooldridge, M. (2002). Game theory and decision theory in multi-agent systems. *Auton. Agent Multi. Agent Syst.* 5, 243–254. doi: 10.1023/A:1015575522401

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). "Curiosity-driven exploration by self-supervised prediction," in *International Conference On Machine Learning (Icml)* (Sydney, NSW), 488–489.

Rao, A. S., and Georgeff, M. P. (1991). "Modeling rational agents within a Bdi-architecture," in *KR'91 Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning* (Cambridge) 91, 473–484.

Roohi, S., Takatalo, J., Guckelsberger, C., and and, H., P. (2018). "Review of intrinsic motivation in simulation-based game testing," in *Proceedings Of The 2018 Chi Conference On Human Factors In Computing Systems* (Montreal, QC: Acm).

Russell, S. J., and Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*. Malaysia: Pearson Education Limited.

Ryan, R. M., and Deci, E. L. (2000a). Intrinsic and extrinsic motivations: classic definitions and new directions. *Contemp. Educ. Psychol.* 25, 54–67. doi: 10.1006/ceps.1999.1020

Ryan, R. M., and Deci, E. L. (2000b). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *Am. Psychol.* 55, 68–78. doi: 10.1037/0003-066X.55.1.68

Salgado, R., Prieto, A., Caamaño, P., Bellas, F., and Duro, R. J. (2016). "Motiven: motivational engine with sub-goal identification for autonomous robots," in *Evolutionary Computation (Cec), 2016 IEEE Congress On, IEEE* (Vancouver, BC), 4887–4894.

Salge, C., Glackin, C., and Polani, D. (2014). Changing the environment based on empowerment as intrinsic motivation. *Entropy* 16, 2789–2819. doi: 10.3390/e16052789

Santucci, V., Baldassarre, G., and Mirolli, M. (2013). Which is the best intrinsic motivation signal for learning multiple skills? *Front. Neurorobot.* 7:22. doi: 10.3389/fnbot.2013.00022

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2016). Grail: a goal-discovering robotic architecture for intrinsically-motivated learning. *IEEE Trans. Cognit. Dev. Syst.* 8, 214–231. doi: 10.1109/TCDS.2016.2538961

Saunders, R., and Gero, J. S. (2004). Curious agents and situated design evaluations. *Ai EDAM* 18, 153–161. doi: 10.1017/S0890060404040119

Savage, T. (2000). Artificial motives: a review of motivation in artificial creatures. *Conn. Sci.* 12, 211–277. doi: 10.1080/095400900750060131

Schembri, M., Mirolli, M., and Baldassarre, G. (2007). "Evolution and learning in an intrinsically motivated reinforcement learning robot," in *European Conference On Artificial Life* (Lisbon: Springer), 294–303.

Schmidhuber, J. (1991). "Curious Model-building Control Systems," in *IEEE International Conference on Neural Networks* (Singapore), 1458–1463.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Trans. Auton. Ment. Dev.* 2, 230–247. doi: 10.1109/TAMD.2010.2056368

Sequeira, P., Melo, F. S., and Paiva, A., (2011). "Emotion-based intrinsic motivation for reinforcement learning, agents," in *International Conference On Affective Computing And Intelligent Interaction* (Berlin; Heidelberg: Springer), 326–336. doi: 10.1007/978-3-642-24600-5_36

Shafi, K., Merrick, K. E., and Debie, E. (2012). "Evolution of intrinsic motives in multi-agent simulations," in *Asia-Pacific Conference On Simulated Evolution And Learning* (Hanoi: Springer), 198–207.

Sigaud, O., and Droniou, A. (2016). Towards deep developmental learning. *IEEE Trans. Cognit. Dev. Syst.* 8, 99–114. doi: 10.1109/TAMD.2015.2496248

Simşek, Ö., and Barto, A. G. (2006). "An intrinsic reward mechanism for efficient exploration," in *Icml 2006 - Proceedings Of The 23rd International Conference On Machine Learning* (New York, NY), 833–840.

Singh, S., Lewis, R. L., Barto, A. G., and Sorg, J. (2010). Intrinsically motivated reinforcement learning: an evolutionary perspective. *IEEE Trans. Auton. Ment. Dev.* 2, 70–82. doi: 10.1109/TAMD.2010.2051031

Sorg, J., Singh, S. P., and Lewis, R. L. (2010). "Internal rewards mitigate agent boundedness," in *Proceedings Of The 27th International Conference On Machine Learning (Icml-10)* (Haifa), 1007–1014.

Stafford, T., Walton, T., Hetherington, L., Thirkettle, M., Gurney, K., and Redgrave, P. (2013). "A novel behavioural task for researching intrinsic motivations," in *Intrinsically Motivated Learning In Natural And Artificial Systems,* eds G. Baldassarre and M. Mirolli (Berlin: Springer), 395–410.

Stone, P., and Veloso, M. (2000). multiagent systems: a survey from a machine learning perspective. *Auton. Robots* 8, 345–383. doi: 10.1023/A:1008942012299

Taffoni, F., Formica, D., Schiavone, G., Scorcia, M., Tomassetti, A., Di Sorrentino, E. P., et al. (2013). "The "Mechatronic Board": a tool to study intrinsic motivations in humans, monkeys, and humanoid robots," in *Intrinsically Motivated Learning In Natural And Artificial Systems,* eds G. Baldassarre and M. Mirolli (Berlin: Springer), 411–432.

Thomaz, A. L., and Breazeal, C. (2006). "Transparency and socially guided machine learning," in *5th International Conference On Development And Learning (Icdl)* (Bloomington).

Thrun, S. (1995). *Exploration in Active Learning*. Cambridge, MA: Handbook Of Brain Science And Neural Networks, 381–384.

Toates, F. M. (1986). *Motivational Systems*. Cambridge: Cambridge University Press.

Watts, A. G., and Swanson, L. W. (2002). *Anatomy Of Motivation, Stevens' Handbook Of Experimental Psychology*. New York, NY: John Wiley & Sons. doi: 10.1002/0471214426.pas0314

White, R. W. (1959). Motivation reconsidered: the concept of competence. *Psychol. Rev.* 66, 297–333. doi: 10.1037/h0040934

Wong, J. M. (2016). *Towards Lifelong Self-Supervision: A Deep Learning Direction for Robotics. arXiv e-prints [Online]*. Available online at: https://ui.adsabs.harvard.edu/#abs/2016arXiv161100201W (Accessed December 10, 2018).

Zhelo, O., Zhang, J., Tai, L., Liu, M., and Burgard, W. (2018). *Curiosity-driven Exploration for Mapless Navigation with Deep Reinforcement Learning. arXiv e-prints [Online]*. Available online at: https://ui.adsabs.harvard.edu/#abs/2018arXiv180400456Z (Accessed April 01, 2018).

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 08 January 2019
doi: 10.3389/fnbot.2018.00087

Learning a Set of Interrelated Tasks by Using a Succession of Motor Policies for a Socially Guided Intrinsically Motivated Learner

Nicolas Duminy[1]*, *Sao Mai Nguyen*[2]* and *Dominique Duhaut*[1]

[1] Département Mathématiques Informatique Statistique, Université Bretagne-Sud, CNRS, Lab-STICC, Lorient, France, [2] IMT Atlantique, Lab-STICC, Université Bretagne Loire (UBL), Nantes, France

We aim at a robot capable to learn sequences of actions to achieve a field of complex tasks. In this paper, we are considering the learning of a set of interrelated complex tasks hierarchically organized. To learn this high-dimensional mapping between a continuous high-dimensional space of tasks and an infinite dimensional space of unbounded sequences of actions, we introduce a new framework called "procedures", which enables the autonomous discovery of how to combine previously learned skills in order to learn increasingly complex combinations of motor policies. We propose an active learning algorithmic architecture, capable of organizing its learning process in order to achieve a field of complex tasks by learning sequences of primitive motor policies. Based on heuristics of active imitation learning, goal-babbling and strategic learning using intrinsic motivation, our algorithmic architecture leverages our procedures framework to actively decide during its learning process which outcome to focus on and which exploration strategy to apply. We show on a simulated environment that our new architecture is capable of tackling the learning of complex motor policies by adapting the complexity of its policies to the task at hand. We also show that our "procedures" enable the learning agent to discover the task hierarchy and exploit his experience of previously learned skills to learn new complex tasks.

Keywords: intrinsic motivation, goal-babbling, multi-task learning, interactive learning, active learning, active imitation learning, hierarchical learning, procedures

Edited by:
Gianluca Baldassarre,
Italian National Research Council, Italy

Reviewed by:
Vishwanathan Mohan,
University of Essex, United Kingdom
Francisco Bellas,
University of A Coruña, Spain
Tara Julia Hamilton,
Macquarie University, Australia

Correspondence:
Nicolas Duminy
nicolas.duminy@telecom-bretagne.eu
Sao Mai Nguyen
nguyensmai@gmail.com

Received: *05 July 2018*
Accepted: *12 December 2018*
Published: *08 January 2019*

Citation:
Duminy N, Nguyen SM and Duhaut D (2019) Learning a Set of Interrelated Tasks by Using a Succession of Motor Policies for a Socially Guided Intrinsically Motivated Learner. Front. Neurorobot. 12:87. doi: 10.3389/fnbot.2018.00087

1. INTRODUCTION

Recently, efforts in the robotic industry and academic field have been made for integrating robots in previously human only environments. In such a context, the ability for service robots to continuously learn new tasks, autonomously or guided by their human counterparts, has become necessary. They would be needed to carry out multiple tasks, especially in open environments, which is still an ongoing challenge in robotic learning. The range of tasks those robots need to learn can be wide and even change after the deployment of the robot. These tasks can also require the execution of complex policies, such as sequences of primitive policies.

Learning to associate a potentially unbounded sequence of policies to a set of infinite tasks is a challenging problem for multi-task learning, because of the high-dimensionality of the policy and state spaces, of multi-task learning, and of the unbounded, continuous and loosely specified environments.

To address these challenges, we examine methods for robots to learn motor policy sequences, methods for multi-task learning, as well as heuristics for learning in high-dimensional mappings such as active learning based on intrinsic motivation, social guidance and strategic learning.

1.1. Learning Motor Policy Sequences

In this article, we tackle the learning of complex policies to complete high-level tasks. More concretely, in this study, we define the policies as a sequence of primitive policies. As we wish to get rid of any a priori on the maximum complexity of the policy needed to complete any task, the sequence of primitive policies can be unbounded. The learning agent thus learns to associate to any outcome or effect on the world, an a priori unbounded sequence of primitive policies. We review in this paragraph works in compositionally of primitives from the robot learning perspective.

A first approach to learning motor policies is to use via-points such as in Stulp and Schaal (2011), Reinhart (2017) or parameterized skills such as in da Silva et al. (2012). The number of via-points or parameters is a way to define the level of complexity of the policies, but these works use a fixed and finite number of via-points. A small number of via-points can limit the complexity of the policies available to the learning agent, while a high number can increase the number of parameters to be learned. Another approach is chain primitive actions into sequences of policies. However, this would increase the difficulty for the learner to tackle simpler tasks which would be reachable using less complex policies. Enabling the learner to decide autonomously the complexity of the policy necessary to solve a task would allow the approach to be adaptive, and suitable to a greater number of problems.

Options Sutton et al. (1999) and (Machado et al., 2017) introduced in the reinforcement learning framework (Sutton and Barto, 1998) offer temporally abstract actions to the learner. These options represent a temporal abstraction of policies as explained in Sutton (2006). Chains of options have been proposed as extensions in order to reach a given target event. Learning simple skills and planning sequences of policies instead of learning a sequence directly has been shown to simplify the learning problem in Konidaris and Barto (2009). They are a way to represent policy probability density in a goal-oriented way. However, each option is built to reach one particular task and they have only been tested for discrete tasks and actions, in which a bounded number of options were used. We would like to reuse this idea of temporal abstraction and goal-oriented representation to create unbounded policy sequences.

1.2. Multi-Task Learning by a Hierarchical Representation

Indeed, an essential component of autonomous, flexible and adaptive robots will be to exploit temporal abstractions, i.e., to treat complex tasks of extended duration, that is to treat complex tasks of extended duration (e.g., making a drawing) not as a single skill, but rather as a sequential combination of skills (e.g., grasping the pen, moving the pen to the initial position of the drawing, etc.) Such task decompositions drastically reduce the search space for planning and control, and are fundamental to making complex tasks amenable to learning. This idea can be traced back to the hypothesis posed in Elman (1993) that the learning needs to be progressive and develop, *starting small*. It has been renamed as *curriculum learning* in Bengio et al. (2009), as formalized in terms of order of the training dataset: the examples should not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones. For multi-task learning in the reinforcement framework, it has been studied as *hierarchical reinforcement learning* as introduced in Barto and Mahadevan (2003), relying on task decomposition or task hierarchy.

Indeed, the relationships between tasks in task hierarchy in Forestier and Oudeyer (2016); Reinhart (2017) have been successfully exploited for learning tool use or learning inverse models for parameterized motion primitives, allowing the robot to reuse previously learned tasks to build more complex ones. As opposed to classical methods enabling robots to learn tool-use, as (Brown and Sammut, 2012) or (Schillaci et al., 2012), which consider tools as objects with affordances to learn using a symbolic representation, (Forestier and Oudeyer, 2016) does not necessitate this formalism and learns tool-use using simply parameterized skills, leveraging on a pre-defined task hierarchy. Barto et al. (2013) showed that building complex actions made of lower-level actions according to the task hierarchy can bootstrap exploration by reaching interesting outcomes more rapidly. Temporal abstraction has also proven to enhance the learning efficiency of a deep reinforcement learner in Kulkarni et al. (2016).

On a different approach (Arie et al., 2012) also showed composing primitive actions through observation of a human teacher enables a robot to build sequences of actions in order to perform object manipulation tasks. This approach relies on neuroscience modeling of mirror neuron systems. From the computational neuroscience point of view for sequence-learning task with trial-and- error, Hikosaka et al. (1999) suggested that procedural learning proceeds as a gradual transition from a spatial sequence to a motor, based on observations that the brain uses two parallel learning processes to learn action sequences: spatial sequence (goal-oriented, task space) mechanism and motor sequence (policy space) mechanism. Each of the acquired motor sequences can also be used as an element of a more-complex sequence.

We would like to extend these ideas of representations of tasks as temporal abstraction and as hierarchies, and to exploit the dual representation of tasks and actions sequences in this paper. Instead of a pre-defined task hierarchy given by the programmer, our robot learner should be able to learn hierarchical representations of its task space to more easily use acquired skills for higher-level tasks.

1.3. Active Motor Learning in High-Dimensional Spaces

In order to learn sequences of primitive policies for multi-task learning, beyond the specific methods for learning sequences of policies and multi-task learning, we would like to review the methods for learning high-dimensional mappings. More specifically, while the cited works above have outlined the importance of the organization and order of the training data, we would like to examine how this organization can be decided online by the robot learner during its learning process, instead of being left to the designer or programmer.

To address the challenge of multi-task motor learning, we will take the point of view of continual learning, also named life-long or curriculum Bengio et al. (2009) learning, that constructs a sophisticated understanding of the world from its own experience to apply previously learned knowledge and skills to new situation with more complex skills and knowledge. Humans and other biological species have this ability to learn continuously from experience and use these as the foundation for later learning. Reinforcement learning, as described in Sutton and Barto (1998), has introduced in a framework for learning motor policies from experience by autonomous data sampling through exploration. However, classical techniques based on reinforcement learning such as Peters and Schaal (2008) and Stulp and Schaal (2011) still need an engineer to manually design a reward function for each particular task, limiting their capability for multi-task learning.

1.3.1. Intrinsic Motivation

More recent algorithms have tried to replace this manually defined reward function, and have proposed algorithms using intrinsic reward, using inspiration from *intrinsic motivation*, which is first described in developmental psychology as triggering curiosity in human beings Deci and Ryan (1985), and has more recently been described in terms of neural mechanisms for information-seeking behaviors (Gottlieb et al., 2013). This theory tries to explain our ability to learn continuously, although we do not have a clear tangible goal other than survival and reproduction, intrinsically motivated agents are still able to learn a wide variety of tasks and specialize in some tasks influenced by their environment and development, even in some tasks that are not directly useful for survival and reproduction. Psychological theories such as intrinsic motivation have tried to explain these apparently non-rewarding behaviors and have been successfully inspired learning algorithms (Oudeyer et al., 2007; Schmidhuber, 2010). More recently, these algorithms have been applied for multi-task learning and have successfully driven the learner's exploration through goal-oriented exploration as illustrated in Baranes and Oudeyer (2010) and Rolf et al. (2010). Santucci et al. (2016) has also proposed a goal-discovering robotic architecture for intrinsically-motivated learning to discover goals and learn corresponding policies, providing the number of goals is preset. Intrinsic motivation has also been coupled with deep reinforcement learning in Colas et al. (2018) to solve sparse or deceptive reward problems to reach a single goal.

However for multi-task learning, especially when the dimension of the outcome space increases, these methods become less efficient (Baranes and Oudeyer, 2013) due to the curse of dimensionality, or when the reachable space of the robot is small compared to its environment. To enable robots to learn a wide range of tasks, and even an infinite number of tasks defined in a continuous space, heuristics such as social guidance can help by driving its exploration toward interesting and reachable space fast.

1.3.2. Social Guidance

Indeed, imitation learning has proven very efficient for learning in high-dimensional space as demonstration can orient the learner toward efficient subspaces. Information could be provided to the robot using external reinforcement signals (Thomaz and Breazeal, 2008), actions (Grollman and Jenkins, 2010), advice operators (Argall et al., 2008), or disambiguation among actions (Chernova and Veloso, 2009). Furthermore, tutors' demonstrations can be combined with autonomous robot learning for more efficient exploration in the sensori-motor space. Initial human demonstrations have successfully initiated reinforcement learning in Muelling et al. (2010) and Reinhart (2017). Nguyen et al. (2011) has combined demonstrations with intrinsic motivation throughout the learning process and shown that autonomous exploration is bootstrapped by demonstrations, enabling the learner to learn mappings in higher-dimensional spaces. Another advantage of introducing imitation learning techniques is to include non-robotic experts in the learning process (Chernova and Veloso, 2009).

Furthermore, tutor's guidance has been shown to be more efficient if the learner can actively request a human for help when needed instead of being passive, both from the learner or the teacher perspective (Cakmak et al., 2010). This approach is called interactive learning and it enables a learner to benefit from both local exploration and learning from demonstration. One of the key elements of these hybrid approaches is to choose when to request human information or learn in autonomy so as to diminish the teacher's attendance.

1.3.3. Strategic Learning

This principle of a learner deciding on its learning process is generalized as *strategic learning*, as formalized in Lopes and Oudeyer (2012). Simple versions have enabled the learner to choose which task space to focus on (Baranes and Oudeyer, 2010), or to to change its strategy online (Baram et al., 2004). In Nguyen and Oudeyer (2012), the algorithm SGIM-ACTS enabled the robot learner to both choose its strategy and target outcome. Owing to its ability to organize its learning process, by choosing actively both which strategy to use and which outcome to focus on, . They have introduced the notion of *strategy* as a method of generating actions and outcome samples This study considered 2 kinds of strategy: autonomous exploration driven by intrinsic motivation and imitation of one of the available human teachers. The SGIM-ACTS algorithm relies on the empirical evaluation of its learning progress. It showed its potential to learn on a real high dimensional robot a set of hierarchically organized tasks in Duminy et al. (2016). This is why we consider to extend SGIM-ACTS to learn to associate a large number of tasks to motor policy sequences.

However, these works have considered a policy space at fixed dimensionality, thus policies of bounded complexity. We would like to extend these methods for unbounded sequences of motor primitives and for larger outcome spaces.

1.4. Can Tutors' Demonstrations Help Learn Sequences of Policies and Task Hierarchies?

In this work in multi-task learning, we want to enable a robot learner to achieve a wide range of tasks that can be inter-related and complex. Based on the state of the art, we base our framework on a parallel representation of sequences of policies as temporal abstraction and sequences of outcomes, as well as a representation of task hierarchies. We will use heuristics of intrinsic motivation, social guidance and strategic learning to enable the robot to learn the high-dimensional mapping between sequences of primitive policies and outcomes, via the learning of the task hierarchies. Thus we will propose a framework for representing hierarchical relationships between tasks and propose a learning algorithm that enables the emergence of such a representation. We will examine the performance of our robot learner for all the tasks defined in the experimental setups according to the point of view of multi-task learning, and we will examine more precisely the performance of our robot learner in the most complex tasks to assess whether it was able to increase its skills. We allow the robot to use sequences of actions of undetermined length to achieve these tasks. The learning algorithm has to face the problem of unlearnability of infinite task and policy spaces, and the curse of dimensionality of sequences of high-dimensionality policy spaces. We developed in Duminy et al. (2018) a new framework called "procedures" (see section 2.2) which proposes to combine known policies represented in a goal-oriented way. This framework showed its ability to improve the learning process of autonomous learners in preliminary experiments.

In this article, we would like to confirm these results by statistical analysis, and most of all, show that interactive strategies can further bootstrap the learning process of such autonomous learners and to help the robot to learn a relevant representation of the hierarchies of tasks. In the next section, we detail our methods based on the procedures framework and the proposed learning algorithm. We will describe in section 3 an experiment, on which we have tested our algorithm, and we will present and analyze the results in section 4.

2. OUR APPROACH

Inspired by developmental psychology, we combine interactive learning and autonomous exploration in a strategic learner, which learning process is driven by intrinsic motivation. This learner also takes task hierarchy into account to reuse its previously learned tasks while adapting the complexity of its policy sequence to the complexity of the task at hand.

In this section, we formalize our learning problem, introduce a goal-oriented representation of sequence of policies and explain the principles of the algorithm SGIM-PB, which is an extension of SGIM-ACTS for learning motor policy sequences of unlimited size. Then, combining it with this "procedures" framework, we developed a new algorithm called Socially Guided Intrinsic Motivation with Procedure Babbling (SGIM-PB) capable of determining a task hierarchy representation to learn a set of complex interrelated tasks using adapted policy sequences.

2.1. Problem Formalization

In our approach, an agent can perform motions through the use of primitive policies π_θ, parameterized by $\theta \in \mathcal{P} \subset \mathbb{R}^n$. It can also perform policy sequences, which are potentially unbounded sequences of primitive motor policies executed sequentially. The policy space $\mathcal{P}^{\mathbb{N}} = \cup_{i\in\mathbb{N}}\mathcal{P}^i$ is the combination of all subspaces $\mathcal{P}^i$ corresponding to each number of primitives, and is a continuous space of infinite dimensionality. Those policies have an effect on the environment, which we call the outcome $\omega \in \Omega$. The agent is then to learn the mapping between the policy space $\mathcal{P}^{\mathbb{N}}$ and Ω: it learns to predict the outcome ω of each policy π_θ (the forward model M), but more importantly, it learns which policy to choose for reaching any particular outcome (an inverse model L). The outcomes ω can be of composite nature and thus be split in subspaces $\Omega_i \subset \Omega$ of different dimensionality. Policy sequences are represented by concatenating the parameters of each of its primitive policies in the execution order.

We take the trial and error approach, and suppose that Ω is a metric space, meaning the learner has a means of evaluating a distance between two outcomes $d(\omega_1, \omega_2)$.

2.2. Procedures

As this algorithm tackles the learning of complex hierarchically organized tasks, exploring and exploiting this hierarchy could ease the learning of the more complex tasks. We define procedures as a way to encourage the robot to reuse previously learned tasks, and chain them to build more complex ones. More formally, a procedure is defined as a succession of previously known outcomes $(\omega_1, \omega_2, ..., \omega_n \in \Omega)$ and is noted $(\omega_1, \omega_1, ..., \omega_n)$. The procedure space is thus simply $\Omega^{\mathbb{N}}$. The definition of the procedure space only depends on the outcome space. But the valid procedures, representing the real dependencies between tasks, depend on each application case. Thus the learning agent can explore the procedure space to test which procedures are valid.

Executing a procedure $(\omega_1, \omega_1, ..., \omega_n)$ means building the policy sequence π corresponding to the succession of policies $\pi_i, i \in [\![1, n]\!]$ (potentially policy sequences as well) and execute it (where the π_i reach best the ω_i $\forall i \in [\![1, n]\!]$ respectively). An example illustrates this idea of task hierarchy in **Figure 1**. As the subtasks ω_i are generally unknown from the learner, the procedure is updated before execution (see **Algorithm 1**) to subtasks ω_i' which are the closest tasks reached by the learner (by executing respectively π_1' to π_n'). When the agent selects a procedure to be executed, this latter is only a way to build the policy sequence which will actually be executed. So the agent does not check if the subtasks are actually reached when executing a procedure.

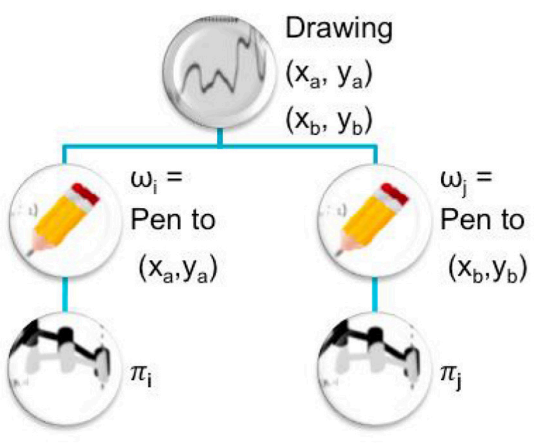

Drawing
(x_a, y_a)
(x_b, y_b)
ω_i = Pen to (x_a,y_a)
ω_j = Pen to (x_b,y_b)
π_i
π_j

FIGURE 1 | Illustration of a procedure or task hierarchy. To make a drawing between points (x_a, y_a) and (x_b, y_b), a robot can recruit subtasks consisting in (ω_i) moving the pen to (x_a, y_a), then (ω_j) moving the pen to (x_b, y_b). These subtasks will be completed respectively with policies π_i and π_j. Therefore to complete the complete this drawing, the learning agent can use the sequence of actions (π_i, π_j).

Algorithm 1 Procedure adaptation

```
Input: $(\omega_1, ..., \omega_n) \in \Omega^n$
Input: inverse model $L$
1: for $i \in [\![1, n]\!]$ do
2:     $\omega_i' \leftarrow$Nearest-Neighbor($\omega_i$) // get the nearest outcome known from $\omega_i$
3:     $\pi_i' \leftarrow L(\omega_i')$ // get the known policy sequence that reached $\omega_i'$
4: end for
5: return $\pi = \pi_1'...\pi_n'$
```

If the procedure given can not be executed by the robot, because at least one of the subtasks space is not reachable, then the procedure is abandoned and replaced by a random policy sequence.

2.3. Socially Guided Intrinsic Motivation With Procedure Babbling

The SGIM-PB algorithm is the last achievement of a series of increasingly complex architectures that we have been developing for autonomous open-ended learning. It is an extension of SGIM-ACTS (Nguyen and Oudeyer, 2012), using the same interest model and memory based inverse model, but it can in addition perform sequences of motor policies. In this study, we limited our learning algorithm to the case of procedures of size 2 (sequences of 2 outcomes only) as we wish to prove the bootstrapping effect of the representation via procedures, before tackling the challenges of exploring a high-dimensional space of procedures $\Omega^{\mathbb{N}}$. This still allows the learning agent to use a high number of subtasks because of the recursivity of the definition of procedures. Our learning algorithm, called SGIM-PB, starts from scratch, it is only provided with the primitive policy space and outcome subspaces dimensionalities and boundaries. The procedural spaces are also predefined, as all the possible composition of outcome subspaces (Ω_i, Ω_j) with $\Omega_i, \Omega_j \subset \Omega$. Then its aim is to learn how to reach a set of outcomes as broad as possible, as fast as possible. This means it has to learn both the possible outcomes to reach and the policy sequences or procedures to use for that. In order to learn, the agent can count on different learning strategies, which are methods to build a policy or procedure from any given target outcome. It also need to map the outcome subspaces and even regions to the best suited strategies to learn them. In this algorithm, the forward and inverse models are memory based and consist only of the cumulative data, mappings of policies, procedures and their respective reached outcomes obtained through all the attempts of the learner. So they are learned by adding new data in the learner's memory.

The SGIM-PB algorithm (see **Algorithm 2**, **Figure 2**) learns by episodes, where it starts by selecting an outcome $\omega_g \in \Omega$ to target and an exploration strategy σ based on its progress as in most competence-based intrinsic motivation algorithms (Baranes and Oudeyer, 2010; Forestier and Oudeyer, 2016) and as detailed in section 2.3.2.

In each episode, the robot starts from the same position before executing a policy, and primitives are executed sequentially without getting back to this initial position. Whole policy sequences are recorded with their outcomes, but each step of the policy sequence execution is also recorded. These data enable the robot to select parts of the policy sequences, thus helping it to optimize the size of policy sequences it executes with respect to the outcomes at hand. The way these data are generated depend on the strategy chosen. We consider two *autonomous exploration* strategies (policy space exploration and procedural space exploration) and two which we call *socially guided exploration* (mimicry of a policy teacher and mimicry of a procedural teacher).

2.3.1. Episodes Using Exploration Strategies

In an episode under the autonomous policy space exploration strategy (line 3 on **Algorithm 2**), the learner tries to optimize the policy π_θ to produce ω_g by choosing between random exploration of policies and local optimization, following the SAGG-RIAC algorithm (Baranes and Oudeyer, 2010) [Goal-Directed Policy Optimization (ω_g)]. This choice is stochastic and based on the closeness of the goal outcome to already known outcomes, local optimization having a higher probability to be selected if the goal outcome neighborhood contains close known outcomes. Random exploration builds a random policy sequence recursively, starting by a random primitive policy, and adding more random primitives according to a probability of $1/\alpha^n$, $\alpha = 2$ being a constant and n the size of the already built policy sequence. Local optimization uses local linear regression. This is a slightly modified version of the SGIM-ACTS autonomous exploration strategy which interpolates from the known policies reaching an outcome close to ω_g.

In an episode under the autonomous procedural space exploration strategy (line 5 on **Algorithm 2**), the learner builds a size 2 procedure (ω_i, ω_j) such as to reproduce the goal outcome ω_g the best using Goal-Directed Optimization [Goal-Directed Procedure Optimization(ω_g)]. The procedure built is then modified and executed, following **Algorithm 1**.

In an episode under the mimicry of a policy teacher strategy (line 7 on **Algorithm 2**), the learner requests a demonstration π_{θ_d} from the chosen teacher. π_{θ_d} is selected by the teacher as the closest from the goal outcome ω_g in its demonstration repertoire. This repertoire is built in advance in practice for our experiments, by recording policies and their reached outcomes. The learner then repeats the demonstrated policy [Mimic Policy (π_{θ_d})]. It is a strategy directly also available in the SGIM-ACTS algorithm.

Algorithm 2 SGIM-PB

```
Input: the different strategies $\sigma_1, ..., \sigma_n$
Initialization: partition of outcome spaces $R \leftarrow \bigsqcup_i \{\Omega_i\}$
Initialization: episodic memory $Memo \leftarrow \emptyset$
 1: loop
 2:   $\omega_g, \sigma \leftarrow$ Select Goal Outcome and Strategy($R$)
 3:   if $\sigma$ = Autonomous policy space exploration strategy then
 4:     $Memo \leftarrow$ Goal-Directed Policy Optimization($\omega_g$)
 5:   else if $\sigma$ = Autonomous procedural space exploration strategy then
 6:     $Memo \leftarrow$ Goal-Directed Procedure Optimization($\omega_g$)
 7:   else if $\sigma$ = Mimicry of policy teacher $i$ strategy then
 8:     $(\pi_{\theta_d}, \omega_d) \leftarrow$ ask and observe demonstrated policy to teacher $i$
 9:     $Memo \leftarrow$ Mimic Policy($\pi_{\theta_d}$)
10:   else if $\sigma$ = Mimicry of procedural teacher $i$ strategy then
11:     $((\omega_{di}, \omega_{dj}), \omega_d) \leftarrow$ ask and observe demonstrated procedure to teacher $i$
12:     $Memo \leftarrow$ Mimic Procedure($(\omega_{di}, \omega_{dj})$)
13:   end if
14:   Update $L$ with collected data $Memo$
15:   $R \leftarrow$ Update Outcome and Strategy Interest Mapping($R$,$Memo$,$\omega_g$)
16: end loop
```

In an episode under the mimicry of a procedural teacher strategy (line 10 on **Algorithm 2**), the learner requests a procedural demonstration of size 2 $(\omega_{di}, \omega_{dj})$ which is built by the chosen teacher according to a preset function which depends on the target outcome ω_g. Then the learner tries to reproduce the demonstrated procedure by refining and executing it, following **Algorithm 1** [Mimic Procedure $(\omega_{di}, \omega_{dj})$].

In both autonomous exploration strategies, the learner uses a method, Goal-Directed Optimization, to optimize its input parameters (procedure for the procedure exploration and policy for the policy exploration) to reach ω_g best. This generic method either creates random inputs, if the goal outcome ω_g is far from any previously reached one, or performs local optimization based on linear regression.

2.3.2. Interest Mapping

After each episode, the learner stores the policies and modified procedures executed along with their reached outcomes in its episodic memory. It computes its *competence* in reaching the goal outcome ω_g by computing the distance $d(\omega_r, \omega_g)$ with the outcome ω_r it actually reached. Then it updates its interest model by computing the interest $interest(\omega, \sigma)$ of the goal outcome and each outcome reached (including the outcome spaces reached but not targeted): $interest(\omega, \sigma) = p(\omega)/K(\sigma)$, where $K(\sigma)$ is the cost of the strategy used and the empirical progress $p(\omega)$ is the difference between the best competence before the attempt and the competence for the current attempt.

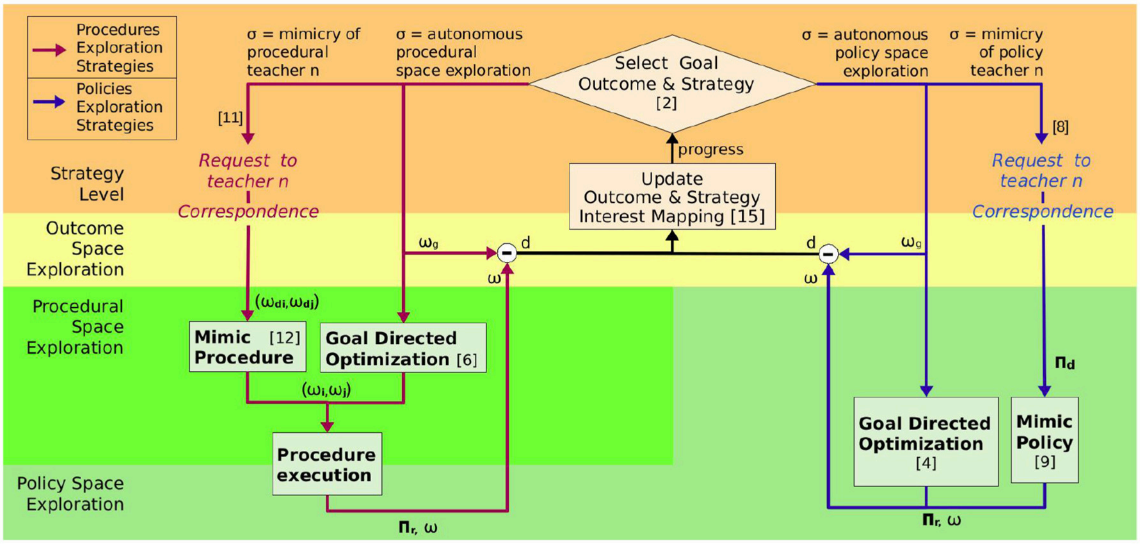

Procedures Exploration Strategies
Policies Exploration Strategies
Strategy Level
Outcome Space Exploration
Procedural Space Exploration
Policy Space Exploration
σ = mimicry of procedural teacher n
σ = autonomous procedural space exploration
Select Goal Outcome & Strategy [2]
σ = autonomous policy space exploration
σ = mimicry of policy teacher n
[11]
[8]
Request to teacher n
Correspondence
progress
Update Outcome & Strategy Interest Mapping [15]
ωg
d
ω
(ωdi,ωdj)
Mimic [12] Procedure
Goal Directed Optimization [6]
(ωi,ωj)
Procedure execution
πr, ω
πd
Goal Directed Optimization [4]
Mimic Policy [9]

FIGURE 2 | SGIM-PB architecture: number between brackets link parts of the architecture with corresponding lines in **Algorithm 2**, the arrows show the data transfer between the different blocks.

The learning agent then uses these interest measures to partition the outcome space Ω into regions of high and low interest. For each strategy σ, the outcomes reached and the goal are added to their partition region. Over a fixed number of measures of interest in the region, it is then partitioned into 2 subregions so as maximize the difference in interest between the 2 subregions. The method used is detailed in Nguyen and Oudeyer (2014). Thus, the learning agent discovers by itself how to organize its learning process and partition its task space into unreachable regions, easy regions and difficult regions, based on empirical measures of interest. This corresponds to line 15 on **Algorithm 2**.

The choice of strategy and goal outcome is based on the empirical progress measured in each region R_n of the outcome space Ω. This corresponds to the line 2 of **Algorithm 2**. ω_g, σ are chosen stochastically (with respectively probabilities p_1, p_2, p_3), by one of the sampling modes:

- mode 1: choose σ and $\omega_g \in \Omega$ at random;
- mode 2: choose an outcome region R_n and a strategy σ with a probability proportional to its interest value. Then generate $\omega_g \in R_n$ at random;
- mode 3: choose σ and R_n like in mode 2, but generate a goal $\omega_g \in R_n$ close to the outcome with the highest measure of progress.

In the start of the learning process, as the robot has no outcome and interest measure to guide this choice, the first mode doing random exploration is automatically selected. At this state, the partition regions consist of the whole outcome subspaces.

The learner can compute nearest neighbors to select policies or procedures to optimize (when choosing local optimization in any of both autonomous exploration strategies and when refining procedures) or when computing the competence to reach a specific goal, it actually uses a performance metric (1) which also takes into account the complexity of the policy chosen:

$$perf(\omega_g) = d(\omega, \omega_g)\gamma^n \tag{1}$$

where $d(\omega, \omega_g)$ is the normalized Euclidean distance between the target outcome ω_g and the outcome ω reached by the policy, γ is a constant and n is equal to the size of the policy (the number of primitives chained).

2.4. Summary

To summarize, we have formalized in this section the problem of multi-task learning as the learning of an inverse model between a composite space of continuous set of outcomes and a space of policies of infinite dimension. The aim is to learn a mapping between outcomes (sometimes referred to as *tasks*) and policy sequences. The learning agent is provided with a set of predefined tasks via a space of outcomes it can observe and a metric to assess the performance of its trials. It can interact with the environment via primitive policies in a predefined space. We then introduced the framework of *procedures* as a goal-directed representation of sequences of primitive policies. To show that procedures can bootstrap the learning of policy sequences, we have proposed SGIM-PB as a learning algorithm that leverages several data collection *strategies* : goal-babbling for autonomous exploration, exploration of procedures, and social guidance to bootstrap the learning. SGIM-PB learns to reach an ensemble of outcomes, by mapping them to policies. As a means, we propose with SGIM-PB to take advantage of the dependencies between tasks. It explores the procedure space to learn these dependencies. Combining these procedures with the learning of simple policies to complete simple tasks, it can build sequences of policies to achieve complex tasks.

We expect the robot to organize its learning process, beginning by learning low-level tasks by exploring the policy space or by imitating the policy teachers. Once it has a good mastery of these low-level tasks, it can take advantage of the dependencies between tasks by exploring the procedural space or imitating the procedural teachers. It thus gradually improves its competence in high-level tasks.

The formalization and algorithmic architecture proposed can apply to multi-task motor learning problems in static environments. The requirements for an experimental setup are:

- to define the primitive policies of the robot in a finite dimensional space.
- to define the different outcomes the user is interested in. This requires (1) defining the variables from the sensors needed and a rough range of their values (we do not need a precise estimation as the algorithm is robust to overestimations of these ranges, see Nguyen and Oudeyer, 2014) (2) a measure for the robot to assess its own performance such as a distance, as in all intrinsic motivation based algorithms. This measure is used as an "internal reward" function. Contrarily to classical reinforcement learning problems, this reward function is not fine tuned to the specific goal at hand, but is a generic function for all the goals in the outcome space. We use a normalized Euclidean distance for all the outcomes in our experiments, in an attempt to show that the specification of tasks for our learning algorithm does not require a fine-tuning as with other reinforcement learning algorithms. We believe that our framework is also applicable to other distance measures. This definition of tasks is probably the most constraining condition, and does not yet scale up well to physical robots in the real world.
- the environment and robot can reset to an initial state, as in most reinforcement learning algorithms.

3. EXPERIMENT

In this study, we designed an experiment with a simulated robotic arm, which can move in its environment and interact with objects in it. We considered a setup with multiple tasks to learn, with tasks independent of each other and tasks that are interdependent. For interdependent tasks, we were inspired by tool use examples such as the setup proposed in Forestier et al. (2017). Our robot can learn an infinite number of tasks, grouped as 6 hierarchically organized types of tasks. The robot is capable of performing policy sequences of unrestricted size (i.e., consisting of any number of primitives), with primitive policies highly redundant and of high dimensionality.

3.1. Simulation Setup

The **Figure 3** shows environmental setup (contained in a cube delimited by $(x, y, z) \in [-1; 1]^3$). The learning agent is a planar robotic arm of 3 joints with the base centered on the horizontal plane, able to rotate freely around the vertical axis (each link has a length of 0.33) and change its vertical position. The robot can grab objects in this environment, by hovering its arm tip (blue in the **Figure 3**) close to them, which position is noted (x_0, y_0, z_0). The robot can interact with:

- Floor (below $z = 0.0$): limits the motions of the robot, slightly elastic which enable the robot to go down to $z = -0.2$ by forcing on it;
- Pen: can be moved around and draw on the floor, broken if forcing too much on the floor (when $z <= -0.3$);
- Joystick 1 (the left one on the figure): can be moved inside a cube-shaped area (automatically released otherwise, position normalized for this area), its x-axis position control a video-game character x position on the screen when grabbed by the robot;
- Joystick 2 (the right one on the figure): can be moved inside a cube-shaped area (automatically released otherwise, position normalized for this area), its y-axis position control a video-game character y position on the screen when grabbed by the robot;
- Video-game character: can be moved on the screen by using the two joysticks, its position is refreshed only at the end of a primitive policy execution for the manipulated joystick.

The robot grabber can only handle one object. When it touches a second object, it breaks, releasing both objects.

The robot always starts from the same position before executing a policy, and primitives are executed sequentially without getting back to this initial position. Whole policy sequences are recorded with their outcomes, but each step of the policy sequence execution is also recorded. This is done so as to enable the robot to select parts of policy sequences when it can, thus helping it to optimize the size of policy sequences it executes with respect to the outcomes at hand.

3.2. Experiment Variables

In this part, we formalize the parameters of the outcome space Ω and the policy space $\mathcal{P}^{\mathbb{N}}$. The distance used to compare two policies together or two outcomes together is the normalized euclidean distance.

3.2.1. Policy Spaces

The motions of each of the three joints of the robot are encoded using a one-dimensional Dynamic Movement Primitive (DMP) which are, as in Pastor et al. (2009), defined by the system:

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(s) \tag{2}$$

$$\tau \dot{x} = v \tag{3}$$

$$\tau \dot{s} = -\alpha s \tag{4}$$

where x and v are the position and velocity of the system; s is the phase of the motion; x_0 and g are the starting and end position of the motion; τ is a factor used to temporally scale the system (set to fix the length of a primitive execution); K and D are the spring constant and damping term fixed for the whole experiment; α is also a constant fixed for the experiment; and f is a non-linear term used to shape the trajectory called the forcing term. This forcing term is defined as:

$$f(s) = \frac{\sum_i w_i \psi_i(s) s}{\sum_i \psi_i(s)} \tag{5}$$

where $\psi_i(s) = \exp(-h_i(s - c_i)^2)$ with centers c_i and widths h_i fixed for all primitives. There are 3 weights w_i per DMP.

The weights of the forcing term and the end positions are the only parameters of the DMP used by the robot. The starting position of a primitive is set by either the initial position of the robot (if it is starting a new policy sequence) or the end position of the preceding primitive. The robot can also set its position on the vertical axis z for every primitive. Therefore, a primitive policy π_θ is parameterized by:

$$\theta = (a_0, a_1, a_2, z) \tag{6}$$

where $a_i = (w_0^{(i)}, w_1^{(i)}, w_2^{(i)}, g^{(i)})$ corresponds to the DMP parameters of the joint i, ordered from base to tip, and z is the fixed vertical position. Thus, the primitive policies space is $\mathcal{P} = \mathbb{R}^{13}$. When combining two or more primitive policies $(\pi_{\theta_0}, \pi_{\theta_1}, ...)$, in a policy sequence π_θ, the parameters $(\theta_0, \theta_1, ...)$ are simply concatenated together from the first primitive to the last. The total policy space, $\mathcal{P} = (\mathbb{R}^{13})^{\mathbb{N}}$ is of unbounded dimension.

3.2.2. Outcome Subspaces

The outcome subspaces the robot learns to reach are hierarchically organized and defined as:

- Ω_0: the position (x_0, y_0, z_0) of the end effector of the robot in Cartesian coordinates at the end of a policy execution;
- Ω_1: the position (x_1, y_1, z_1) of the pen at the end of a policy execution if the pen is grabbed by the robot;
- Ω_2: the first (x_a, y_a) and last (x_b, y_b) points of the last drawn continuous line on the floor if the pen is functional (x_a, y_a, x_b, y_b);
- Ω_3: the position (x_3, y_3, z_3) of the first joystick at the end of a policy execution if it is grabbed by the robot;
- Ω_4: the position (x_4, y_4, z_4) of the second joystick at the end of a policy execution if it is grabbed by the robot;
- Ω_5: the position (x_5, y_5) of the video-game character at the end of a policy execution if moved.

The outcome space is a composite and continuous space $\Omega = \cup_{i=0}^{5} \Omega_i$, with subspaces of 3 to 4 dimensions. A quick analysis of this setup highlights interdependencies between tasks: controlling the position of the pen comes after controlling the position of the end effector; and controlling the position of the video-game character comes after controlling the positions of both joysticks, which in turn comes after controlling the position of the end effector. In our setup, the most complex

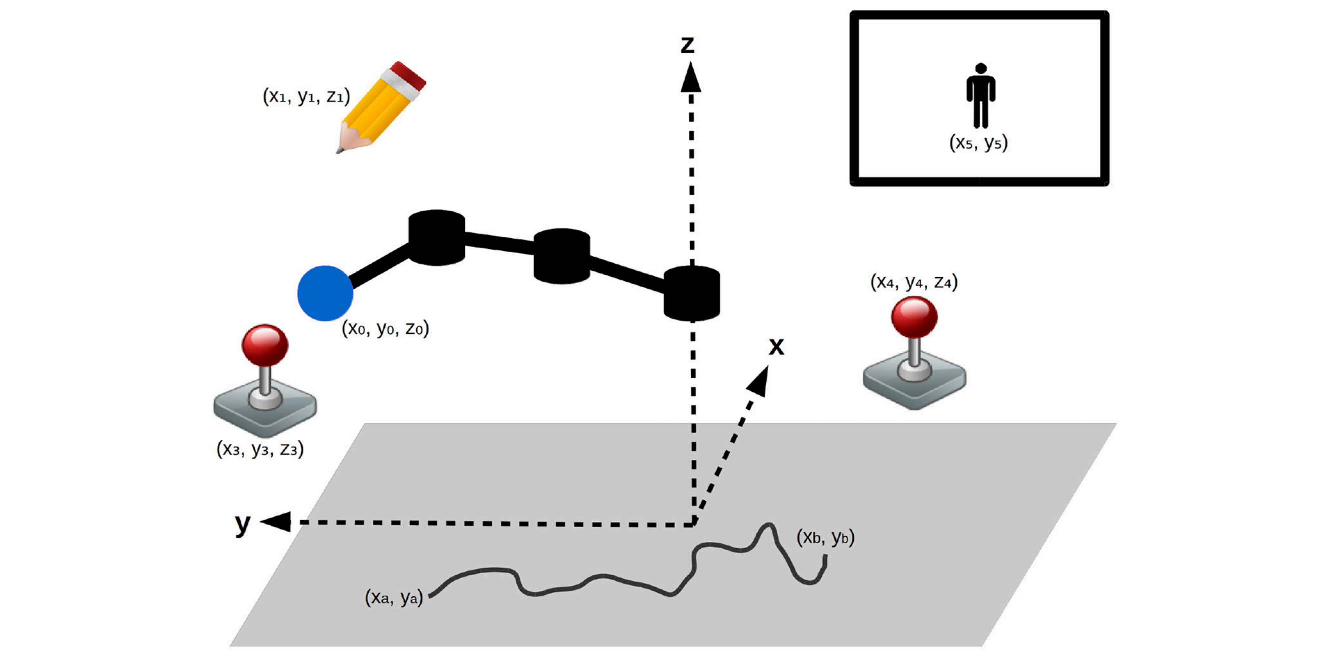


FIGURE 3 | Experimental setup: a robotic arm, can interact with the different objects in its environment (a pen and two joysticks). Both joysticks enable to control a video-game character (represented in top-right corner). A gray floor limits its motions and can be drawn upon using the pen (a possible drawing is represented).

task is controlling the position of the video-game character. This task should require a sequence of 4 actions : move the end-effector to initial position of the joystick 1, move joystick 1, then move the end-effector to the initial position of joystick 2, and move joystick 2. Besides, there are independent tasks: the position of the pen does not really depend on the position of the video-game character. Therefore, the inter-dependencies can be grouped into 2 dependency graphs. With this setup, we test if the procedures found by the robot can distinguish between dependent and independent tasks, and can compose tools uses.

The robot will choose at every episode a goal to reach in the outcome space Ω. In the beginning of its learning process, we expect the robot to make good progress in the easy tasks in Ω_0 then $\Omega_1, \Omega_3, \Omega_4$ using *Autonomous policy space exploration* and *Mimicry of policy teacher* strategies. Once it has a good mastery of the easy tasks, it will concentrate on the more difficult tasks, and will benefit from procedures most, using *Autonomous procedural space exploration* and *Mimicry of procedural teacher* strategies.

In our multi-task learning perspective, we will examine how well the robot performs for each of the tasks in these subspaces. We will particularly examine its performance for the tasks of Ω_5, which we consider the most complex tasks.

3.3. The Teachers

Our SGIM-PB learner can actively learn by asking teachers to give demonstrations of procedures or policies (strategies *Mimic procedural teacher* and *Mimic policy teacher*).

To help the SGIM-PB learner, procedural teachers were available so as to provide procedures for every complex outcome subspaces Ω_1, Ω_2, Ω_3, Ω_4 and Ω_5. As Ω_0 is the simplest outcome space in our setup, the base of its task hierarchy, we decided to build the preset functions for these procedural teachers up from Ω_0. Each teacher was only giving procedures useful for its own outcome space, and was aware of its task representation. When presented with an outcome outside its outcome space of expertise, it provides a demonstration for a newly drawn random target outcome in its outcome space of expertise. They all had a cost of 5. The rules used to provide procedures are the following:

- ProceduralTeacher1 ($\omega_{1_g} \in \Omega_1$): (ω_1, ω_0) with $\omega_1 \in \Omega_1$ equals to the pen initial position and $\omega_0 \in \Omega_0$ equals to the desired final pen position ω_{1_g};
- ProceduralTeacher2 ($\omega_{2_g} = (x_a, y_a, x_b, y_b) \in \Omega_2$): (ω_1, ω_0) with $\omega_1 \in \Omega_1$ equals to the point on the $z = 1.0$ plane above the first point of the desired drawing $\omega_1 = (x_a, y_a, 1)$, and $\omega_0 \in \Omega_0$ equals to the desired final drawing point, $\omega_0 = (x_b, y_b, 0)$;
- ProceduralTeacher3 ($\omega_{3_g} \in \Omega_3$): (ω_3, ω_0) with $\omega_3 = (0, 0, 0), \omega_3 \in \Omega_3$ and $\omega_0 \in \Omega_0$ equals to the end effector position leading to the desired final position of the first joystick ω_{3_g};
- ProceduralTeacher4 ($\omega_{4_g} \in \Omega_4$): (ω_4, ω_0) with $\omega_4 = (0, 0, 0), \omega_4 \in \Omega_4$ and $\omega_0 \in \Omega_0$ equals to the end effector position leading to the desired final position of the second joystick ω_{4_g};
- ProceduralTeacher5 ($\omega_{5_g} = (x, y) \in \Omega_5$): (ω_3, ω_4) with $\omega_3 = (x, 0, 0), \omega_3 \in \Omega_3$ with x corresponding to the desired x-position of the video-game character, $\omega_4 = (0, y, 0), \omega_4 \in$

Ω_4 with y corresponding to the desired y-position of the video-game character.

We also added policy teachers corresponding to the same outcome spaces to bootstrap the robot early learning process. The strategy attached to each teacher has a cost of 10. Each teacher was capable to provide demonstrations (as policies executable by the robot) linearly distributed in its outcome space. All those teachers consist of demonstrations repertoires built by drawing sparse demonstrations from a random policy learner trained a huge amount of time (1,000,000 iterations):

- MimicryTeacher1 (Ω_1): 15 demonstrations;
- MimicryTeacher2 (Ω_2): 25 demonstrations;
- MimicryTeacher3 (Ω_3): 18 demonstrations;
- MimicryTeacher4 (Ω_4): 18 demonstrations;
- MimicryTeacher5 (Ω_5): 9 demonstrations;

These costs were chosen so as to encourage the robot to rely on itself as much as possible to reduce the teacher load. The costs of 10 for a policy teacher strategy and 5 for a procedural teacher are arbitrary. Their difference comes from our belief that giving a procedure takes less time to the teacher than providing it with a detailed demonstrated motor policy.

3.4. Evaluation Method

To evaluate our algorithm, we created a benchmark dataset for each outcome space Ω_i, linearly distributed across the outcome space dimensions, for a total of 27,600 points. The evaluation consists in computing the normalized Euclidean distance between each of the benchmark outcome and their nearest neighbor in the learner dataset. Then we compute the mean distance to benchmark for each outcome space. The global evaluation is the mean evaluation for the 6 outcome spaces. This evaluation is repeated across the learning process at predefined and regularly distributed timestamps.

Then to asses our algorithm efficiency, we compare its results with 3 other algorithms:

- SAGG-RIAC: performs autonomous exploration of the policy space $\mathcal{P}^{\mathbb{N}}$ guided by intrinsic motivation;
- SGIM-ACTS: interactive learner driven by intrinsic motivation. Choosing between autonomous exploration of the policy space $\mathcal{P}^{\mathbb{N}}$ and mimicry of one of the available policy teachers;
- IM-PB: performs both autonomous exploration of the procedural space and the policy space, guided by intrinsic motivation;
- SGIM-PB: interactive learner driven by intrinsic motivation. Choosing between autonomous exploration strategies (either of the policy space or the procedural space) and mimicry of one of the available teachers (either policy or procedural teachers).

For each run for all algorithms, we let the algorithm perform arbitrarily 25,000 iterations (policy sequences executions or learning episodes). The value of γ for this experiment is 1.2. The probabilities to choose either of the sampling mode of SGIM-PB are $p_1 = 0.15$, $p_2 = 0.65$, $p_3 = 0.2$. The code run for this experiment can be found in https://bitbucket.org/smartan117/sgim_iclr.

4. RESULTS

4.1. Distance to Goals

The **Figure 4** shows the global evaluation of all the tested algorithms, which corresponds to the mean error made by each algorithm to reproduce the benchmarks with respect to the number of complete policy sequences tried. Random, SGIM-ACTS, SGIM-PB were run 20 times while IM-PB and SAGG-RIAC was run 10 times on this setup so as to obtain statistically significant differences between SGIM-PB and the other algorithms, according to the Student's t-test on two algorithms : $p = 3 * 10^{-16} < 0.1$ when compared with random, $p = 0.01$ for SAGG-RIAC, $p = 1 * 10^{-9}$ for SGIM-ACTS. The complete results for Student's t-test are reported in **Table A1** in the Annex. The algorithms capable of performing procedures (IM-PB and SGIM-PB) have errors that drop to levels lower than the their non-procedure equivalents (SAGG-RIAC and SGIM-ACTS). The t-test comparing the final errors of IM-PB and SGIM-PB vs. SAGG-RIAC and SGIM-ACTS gives a strong difference with $p = 9e - 4 < 0.1$. Moreover, this difference starts since the beginning of the learning process (shown on **Figure 4**). It seems that the procedures bootstrap the exploration, enabling the learner to progress further. Indeed, the autonomous learner IM-PB learner, the upgraded version of SAGG-RIAC by the use of procedures, has significantly better performance.

We can also see that the SGIM-PB algorithm has a very quick improvement in global evaluation owing to the bootstrapping effect of the different teachers. It goes lower to the final evaluation of SAGG-RIAC (0.17) after only 500 iterations. This bootstrapping effect comes from the mimicry teachers, as it is also observed for SGIM-ACTS which shares the same mimicry teachers.

If we look at the evaluation on each individual outcome space (**Figure 5**), we can see that the learners with demonstrations (SGIM-PB and SGIM-ACTS) outperform the other algorithms, except for the most simple outcome space Ω_0, which does not require sequences of actions, and the outcome space Ω_5. In the case of Ω_5, the difference with IM-PB is not significative (IM-PB seems a bit better but the difference is not significative with $p > 0.1$). The results for Student's t-test are reported in **Table A1** in the Annex. This exception for Ω_5 is due to the fact that IM-PB practiced much more on this outcome space (1500 iterations where it chose goals in Ω_5 against 160 for SGIM-PB). SGIM-PB and SGIM-ACTS are much better than the other algorithms on the two joysticks outcome spaces (Ω_3 and Ω_4) (with respectively p=7e-4 and 1e-5). This is not surprising given the fact that those outcome spaces require precise policies. Indeed, if the end-effector gets out of the area where it can control the joystick, the latter is released, thus potentially ruining the attempt. So on these outcome spaces working directly on carefully crafted policies can alleviate this problem, while using

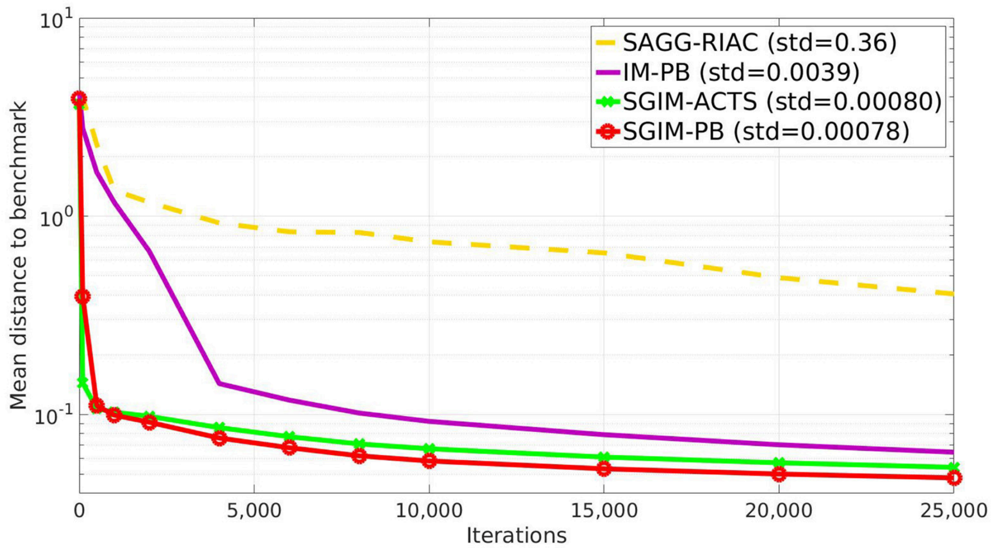


FIGURE 4 | Evaluation of all algorithms (final standard deviation shown in caption).

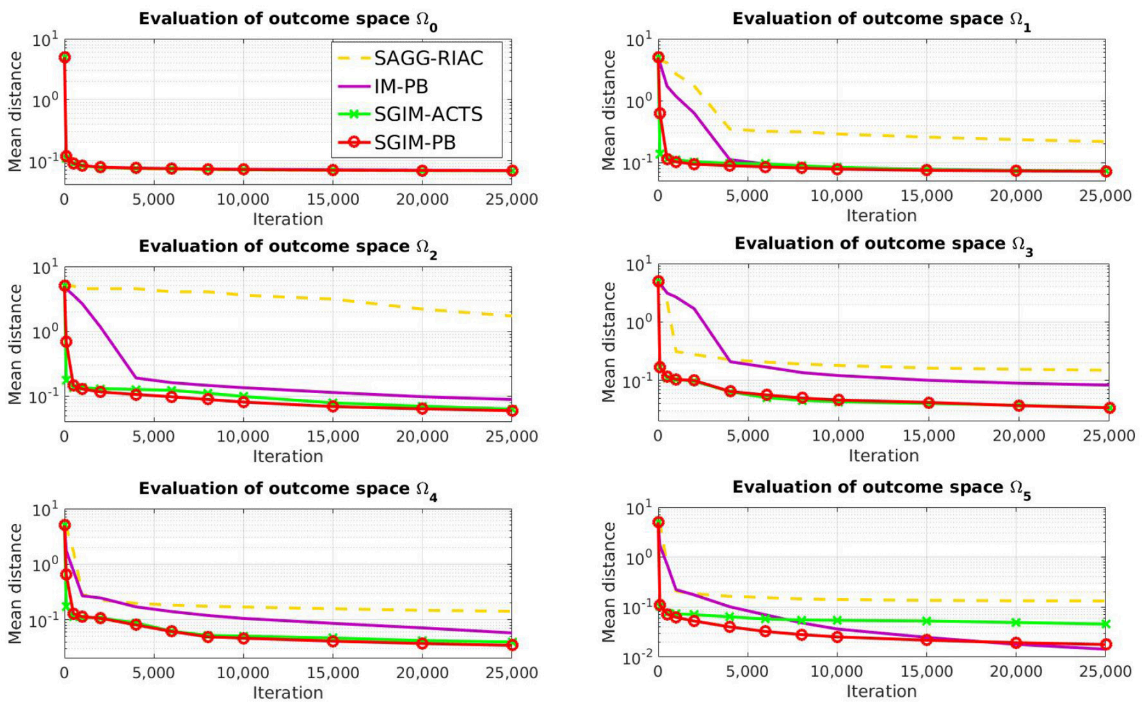


FIGURE 5 | Evaluation of all algorithms per outcome space (for Ω_0, all evaluations are superposed).

procedures might be tricky, as the outcomes used don't take into account the motion trajectory but merely its final state. SGIM-PB was provided with such policies by the policy teachers. Also if we compare the results of the autonomous learner without procedures (SAGG-RIAC) with the one with procedures (IM-PB), we can see that it learns less on any outcome space but Ω_0 (which was the only outcome space reachable using only single primitive policies and that could not benefit from using the task hierarchy to be learned) and especially for Ω_1, Ω_2 and Ω_5 which were the most hierarchical in this setup. More generally, it seems than on this highly hierarchical Ω_5, the learners with procedures were better. So the procedures helped when learning any potentially hierarchical task in this experiment.

4.2. Analysis of the Sampling Strategy Chosen for Each Goal

We further analyzed the results of our SGIM-PB learner. We looked in its learning process to see which pairs of teachers and target outcomes it has chosen (**Figure 6**). It was capable to request demonstrations from the relevant teachers depending on the task at hand, except for the outcome space Ω_0 which had no human teachers and therefore could not find a better teacher to help it. Indeed, for the outcome space Ω_2, the procedural teacher (ProceduralTeacher2) specially built for this outcome space was greatly chosen.

We wanted to see if our SGIM-PB learner adapts the complexity of its policies to the working task. So we looked which policy space would be chosen by the local optimization function (used inside the policy space exploration strategy) for the Ω_0, Ω_1 and Ω_2 subspaces (chosen because they are increasingly complex) on their respective evaluation benchmarks. We compared those results with the same obtained by the IM-PB learner to see if the teachers had an effect on the complexity of policies produced. **Figure 7** shows the results of this analysis.

4.3. Length of the Sequence of Primitive Policies

As we can see on those three interrelated outcome subspaces (**Figure 7**), the learner is capable to adapt the complexity of its policies to the outcome at hand. It chooses longer policies for the Ω_1 subspace (policies of size 2 and 3 while using mostly policies of size 1 and 2 for Ω_0) and even longer for the Ω_2 subspace (using far more policies of size 3 than for the others). It shows that our learner is capable to correctly limit the complexity of its policies instead of being stuck into always trying longer and longer policies. Also, if we look at the policy complexity of the IM-PB learner, we see it was also capable to correctly limit its complexity (especially on Ω_0 where it used even more single-primitive policies than SGIM-PB). However, we can see that our SGIM-PB learner, owing to the teacher strategies available to it, had a smaller spread on the size of policy sequences distribution for each of the three outcome spaces.

We also wanted to see if our SGIM-PB algorithm had discovered the task hierarchy of this experiment. We hoped it would correctly assess which procedural space is adapted to each of the complex outcome subspaces (all subspaces except Ω_0 as it cannot benefit from procedures to be reached). So we looked which procedural space was selected by the local optimization function (used inside the procedural space exploration strategy) for each of the outcome subspaces on their respective evaluation benchmarks. For assessing those results, we compared them with those obtained by the IM-PB learner on the same process.

As we can see on left column of **Figure 8** and **Figure A1**, our SGIM-PB learner successfully chooses the procedural spaces most adapted for each complex outcome subspace (the same as those we used to build the procedural teachers). For instance, to move the video character (task Ω_5), the robot mainly uses subtasks Ω_4 (position of the second joystick) and Ω_3 (position of the first joystick). To move the position of the first joystick (task Ω_3), subtasks Ω_0 (position of the end-effector) and Ω_3 (position of the first joystick) are used. The same way, task Ω_4 recruits subtasks Ω_0 and Ω_4. Thus by recursively, the robot has built a hierarchical representation that task Ω_5 depends on subtasks $(\Omega_0, \Omega_4, \Omega_0, \Omega_3)$. This means it was successfully able to discover and exploit it. By comparison, the IM-PB learner was only capable to identify useful procedural spaces for the Ω_1 and Ω_2 outcome subspaces. For both those outcome subspaces, it identified the one procedural space mainly used by SGIM-PB learner and another one (Ω_2, Ω_0) which can also be useful to learn to reach those outcome subspaces, though arguably less efficient. Indeed, using a policy moving the pen (in Ω_1) is enough for the first component of procedures used to reach Ω_1 and Ω_2, and it can lead to less complex policy sequences than using one drawing on the floor (in Ω_2). If we look at the result for the outcome subspaces Ω_3 and Ω_4, the IM-PB learner was incapable to identify adapted procedural spaces. The absence of a policy teacher to guide it could explain the IM-PB learner poor results on those outcome subspaces. Also, compared to the great focus of the SGIM-PB learner on this outcome subspaces, IM-PB results were more dispersed, indicating its difficulty to select an adapted procedural space. As those outcome subspaces require precise policies and are less adapted to procedures, this difficulty is understandable. By looking at the results of both learners, we can see that the procedural teachers had a profound impact on the choice of adapted procedures for each outcome subspaces, and clearly guided its whole learning process by helping it discover the task hierarchy of the experimental setup.

5. CONCLUSION AND FUTURE WORK

5.1. Conclusion

With this experiment, we show the capability of SGIM-PB to tackle the learning of a set of multiple interrelated complex tasks. It successfully discovers the hierarchy between tasks and uses sequences of motor policies to learn a wider range of tasks. It is capable to correctly choose the most adapted teachers to the target outcome when available. Though it is not limited in the size of policies it could execute, the learner shows it could adapt the complexity of its policies to the task at hand.

The procedures greatly improved the learning capability of autonomous learners, as shows the difference between IM-PB and SAGG-RIAC. Our SGIM-PB shows it is capable to use procedures to discover the task hierarchy and exploit the inverse model of previously learned tasks. More importantly, it shows it can successfully combine the ability of SGIM-ACTS to progress quickly in the beginning (owing to the mimicry teachers) and the ability of IM-PB to progress further on highly hierarchical tasks (owing to the procedure framework).

5.2. Contributions

In this article, we aimed to enable a robot to learn sequences of actions of undetermined length to achieve a field of outcomes. To tackle this high-dimensionality learning between a continuous high-dimensional space of outcomes and a continuous infinite dimensionality space of sequences of actions, we used techniques that have proven efficient in previous studies: goal-babbling, social guidance and strategic learning based on intrinsic

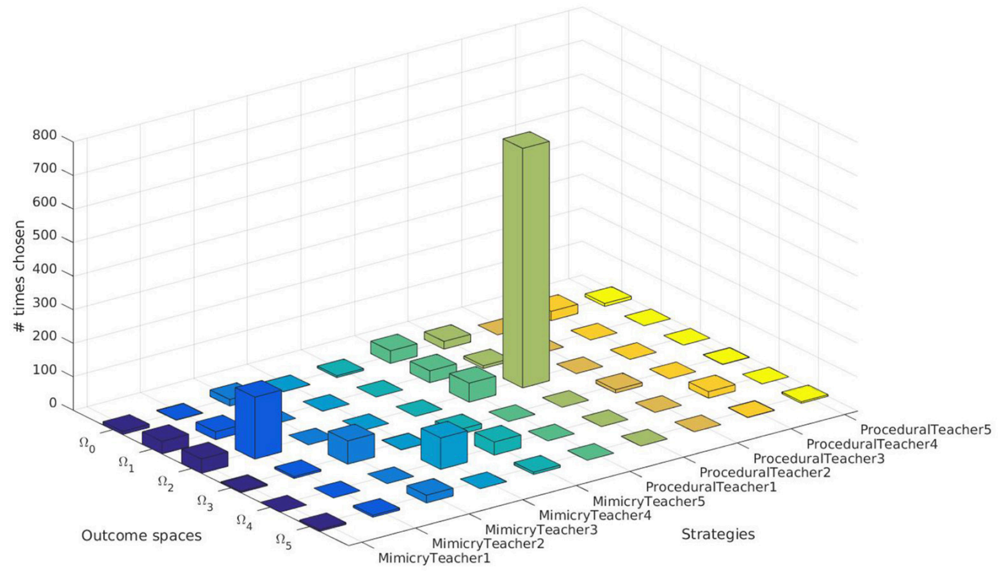


FIGURE 6 | Choices of teachers and target outcomes of the SGIM-PB learner.

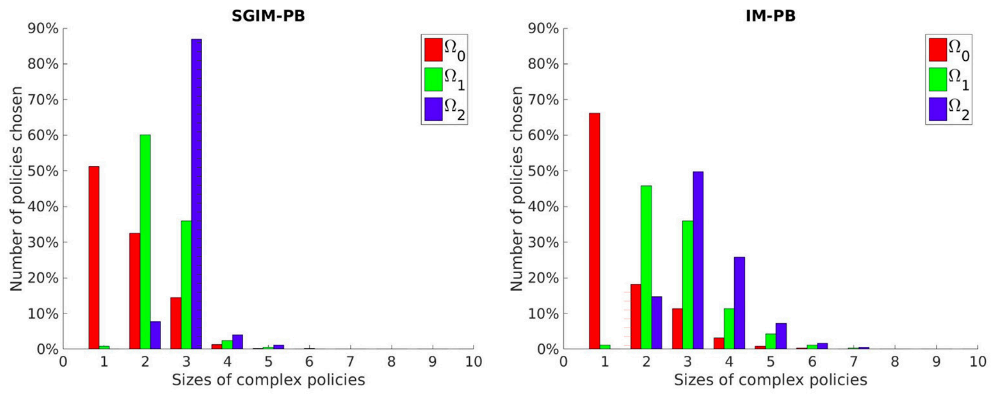


FIGURE 7 | Number of policies selected per policy size for three increasingly more complex outcome spaces by the SGIM-PB (on the left) and IM-PB (on the right) learners.

motivation. We extended them with the procedures framework and proposed SGIM-PB algorithm, allowing the robot to babble in the procedure space and to imitate procedural teachers. We showed that SGIM-PB can discover the hierarchy between tasks, learn to reach complex tasks while adapting the complexity of the policy. Although the representation of actions and tasks are predefined, we described a developmental process involved in the emergence of representations of tasks highlighting their relationships. The study shows that:

- procedures allow the learner to learn complex tasks, and adapt the length of sequences of actions to the complexity of the task
- social guidance bootstraps the learning owing to demonstrations of primitive policy in the beginning, and then to demonstrations of procedures to learn how to compose tasks into sequences of actions
- intrinsic motivation can be used as a common criteria for active learning for the robot to choose both its exploration strategy, its goal outcomes and the goal-oriented procedures.

Our contributions in the field of cognitive robotics are to highlight (1) the relevance of a parallel representation of sequences in the action and the task space, through a goal-oriented temporal abstraction (2) the importance of a hierarchical representation of tasks in multi-task learning problems, and (3) the efficiency of active strategical learning in curriculum learning. An intrinsically motivated robot can learn how to collect data in an organized and meaningful order, from simple to more complex tasks. We have presented a developmental process involved in the emergence of representations of action and tasks.

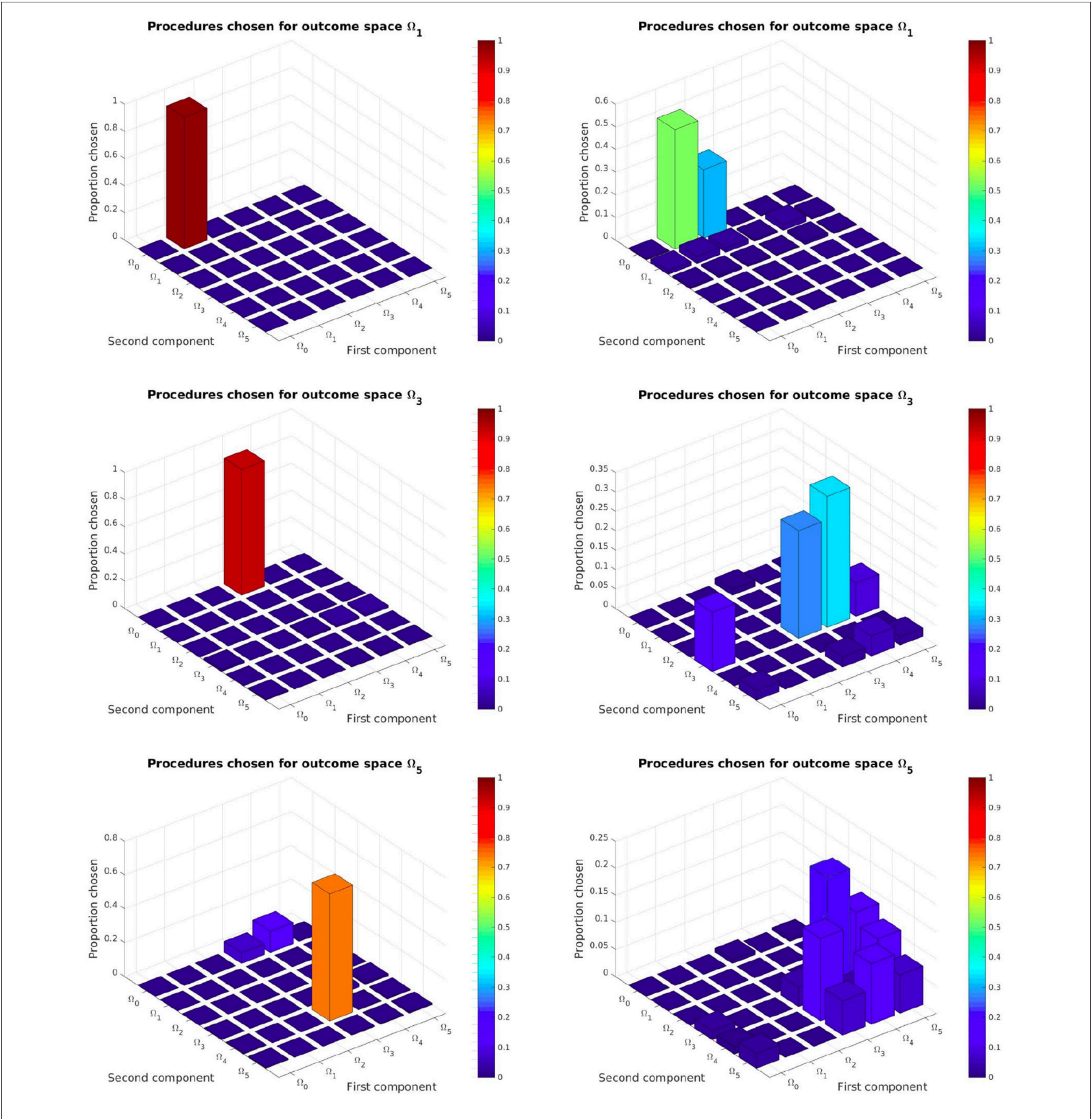


FIGURE 8 | Task hierarchy discovered by the SGIM-PB (left side) and IM-PB (right side) learners for the outcome spaces Ω_1, Ω_3, Ω_5: this represents for each complex outcome space the percentage of time each procedural space would be chosen. See Appendix A for the complete figure on **Figure A1**.

5.3. Future Work

However a precise analysis of the impact of each of the different strategies used by our learning algorithm could give us more insight in the roles of the teachers and procedures framework. Also, we aim to illustrate the potency of our SGIM-PB learner on a real-world application. We are currently designing such an experiment with a physical robotic platform.

Besides, the procedures are defined as combinations of any number of subtasks but the algorithm we submitted only uses procedures as combinations of 2 subtasks. Because of the recursive definition of procedures, the robot can still have representations of complex tasks as composed of numerous subtasks. However, in order to have a direct representation of an unbounded number of tasks, it could be a next step to see if the learning algorithm can handle the curse of dimensionality of a larger procedure space, and explore combinations of any number of subtasks. Moreover, the algorithm can be extended to allow the robot learner to decide on how to execute a procedure. In the current version, we have proposed the "refinement process" to infer the best policy. We could make this refinement process more recursive, by allowing the algorithm to select, not only policies, but also lower-level procedures as one of the policy components.

AUTHOR CONTRIBUTIONS

ND wrote most parts of the paper. SN wrote more specifically parts of the introduction and of the approach parts. She also proposed corrections to the article, as did DD, who also oriented the research as ND thesis director.

ACKNOWLEDGMENTS

This work was supported by public funds [Ministère de l'Education Nationale, de l'Enseignement Supérieur et de la Recherche, European Regional Development Fund (ERDF), Région Bretagne, Conseil Général du Finistère] and by Institut Mines Télécom, received in the framework of the VITAAL project.

This article follows the work in Duminy et al. (2018), extends its algorithm and add more complete analyzes.

REFERENCES

Argall, B. D., Browning, B., and Veloso, M. (2008). "Learning robot motion control with demonstration and advice-operators," in *In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems* (Nice: IEEE), 399–404.

Arie, H., Arakaki, T., Sugano, S., and Tani, J. (2012). Imitating others by composition of primitive actions: a neuro-dynamic model. *Robot. Auton. Syst.* 60, 729–741. doi: 10.1016/j.robot.2011.11.005

Baram, Y., El-Yaniv, R., and Luz, K. (2004). Online choice of active learning algorithms. *J. Mach. Learn. Res.* 5, 255–291.

Baranes, A., and Oudeyer, P.-Y. (2010). "Intrinsically motivated goal exploration for active motor learning in robots: a case study," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Taipei: IEEE), 1766–1773.

Baranes, A., and Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robot. Auton. Syst.* 61, 49–73. doi: 10.1016/j.robot.2012.05.008

Barto, A. G., Konidaris, G., and Vigorito, C. (2013). "Behavioral hierarchy: exploration and representation," in *Computational and Robotic Models of the Hierarchical Organization of Behavior*, eds G. Baldassare and M. Mirolli (Berlin; Heidelberg: Springer), 13–46.

Barto, A. G., and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dyn. Syst.* 13, 41–77. doi: 10.1023/A:1022140919877

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY: ACM), 41–48.

Brown, S., and Sammut, C. (2012). "A relational approach to tool-use learning in robots," in *International Conference on Inductive Logic Programming* (Berlin; Heidelberg: Springer), 1–15.

Cakmak, M., Chao, C., and Thomaz, A. L. (2010). Designing interactions for robot active learners. *IEEE Trans. Auton. Mental Develop.* 2, 108–118. doi: 10.1109/TAMD.2010.2051030

Chernova, S., and Veloso, M. (2009). Interactive policy learning through confidence-based autonomy. *J. Artif. Intellig. Res.* 34:1. doi: 10.1613/jair.2584

Colas, C., Sigaud, O., and Oudeyer, P.-Y. (2018). Gep-pg: decoupling exploration and exploitation in deep reinforcement learning algorithms. *arXiv preprint arXiv:1802.05054.*

da Silva, B., Konidaris, G., and Barto, A. G. (2012). "Learning parameterized skills," in *29th International Conference on Machine Learning (ICML 2012)* (Edinburgh).

Deci, E., and Ryan, R. M. (1985). *Intrinsic Motivation and Self-Determination in Human Behavior*. New York, NY: Plenum Press.

Duminy, N., Nguyen, S. M., and Duhaut, D. (2016). "Strategic and interactive learning of a hierarchical set of tasks by the Poppy humanoid robot," in *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)* (Cergy-Pontoise), 204–209.

Duminy, N., Nguyen, S. M., and Duhaut, D. (2018). "Learning a set of interrelated tasks by using sequences of motor policies for a strategic intrinsically motivated learner," in *IEEE International Robotics Conference* (Laguna Hills, CA).

Elman, J. (1993). Learning and development in neural networks: the importance of starting small. *Cognition* 48, 71–99. doi: 10.1016/0010-0277(93)90058-4

Forestier, S., Mollard, Y., and Oudeyer, P.-Y. (2017). Intrinsically motivated goal exploration processes with automatic curriculum learning. *CoRR*. abs/1708.02190.

Forestier, S., and Oudeyer, P.-Y. (2016). "Curiosity-driven development of tool use precursors: a computational model," in *38th Annual Conference of the Cognitive Science Society (CogSci 2016)* (Philadelphia, PA), 1859–1864.

Gottlieb, J., Oudeyer, P.-Y., Lopes, M., and Baranes, A. (2013). Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends Cogn. Sci.* 17, 585–593. doi: 10.1016/j.tics.2013.09.001

Grollman, D. H., and Jenkins, O. C. (2010). "Incremental learning of subtasks from unsegmented demonstration," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Taipei: IEEE), 261–266.

Hikosaka, O., Nakahara, H., Rand, M. K., Sakai, K., Lu, X., Nakamura, K., et al. (1999). Parallel neural networks for learning sequential procedures. *Trends Neurosci.* 22, 464–471. doi: 10.1016/S0166-2236(99)01439-3

Konidaris, G., and Barto, A. G. (2009). "Skill discovery in continuous reinforcement learning domains using skill chaining," in *Advances in Neural Information Processing Systems (NIPS)* (Vancouver, BC), 1015–1023.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. (2016). "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in Neural Information Processing Systems* (Barcelona), 3675–3683.

Lopes, M., and Oudeyer, P.-Y. (2012). "The strategic student approach for life-long exploration and learning," in *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)* (San Diego, CA: IEEE), 1–8.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2017). A laplacian framework for option discovery in reinforcement learning. *arXiv preprint arXiv:1703.00956.*

Muelling, K., Kober, J., and Peters, J. (2010). "Learning table tennis with a mixture of motor primitives," in *2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (Nashville, TN: IEEE), 411–416.

Nguyen, S. M., Baranes, A., and Oudeyer, P.-Y. (2011). "Bootstrapping intrinsically motivated learning with human demonstrations," in *IEEE International Conference on Development and Learning*, Vol. 2 (Trondheim: IEEE) 1–8.

Nguyen, S. M., and Oudeyer, P.-Y. (2012). Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner. *Paladyn J. Behav. Robot.* 3, 136–146. doi: 10.2478/s13230-013-0110-z

Nguyen, S. M., and Oudeyer, P.-Y. (2014). Socially guided intrinsic motivation for robot learning of motor skills. *Auton. Robots* 36, 273–294. doi: 10.1007/s10514-013-9339-y

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evol. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *2009. ICRA'09. IEEE International Conference on Robotics and Automation* (Kobe: IEEE), 763–768.

Peters, J., and Schaal, S. (2008). Natural actor critic. *Neurocomputing* 71, 1180–1190. doi: 10.1016/j.neucom.2007.11.026

Reinhart, R. F. (2017). Autonomous exploration of motor skills by skill babbling. *Auton. Robots* 41, 1521–1537. doi: 10.1007/s10514-016-9613-x

Rolf, M., Steil, J., and Gienger, M. (2010). Goal babbling permits direct learning of inverse kinematics. *IEEE Trans. Auton. Mental Develop.* 2, 216–229. doi: 10.1109/TAMD.2010.2062511

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2016). Grail: a goal-discovering robotic architecture for intrinsically-motivated learning. *IEEE Trans. Cogn. Develop. Syst.* 8, 214–231. doi: 10.1109/TCDS.2016.2538961

Schillaci, G., Hafner, V. V., and Lara, B. (2012). "Coupled inverse-forward models for action execution leading to tool-use in a humanoid robot," in *Proceedings of the seventh annual ACM/IEEE International Conference on Human-Robot Interaction* (Boston, MA: ACM), 231–232.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Trans. Auton. Mental Develop.* 2, 230–247. doi: 10.1109/TAMD.2010.2056368

Stulp, F., and Schaal, S. (2011). "Hierarchical reinforcement learning with movement primitives," in *2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (Bled: IEEE), 231–238.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning: an Introduction*. Cambridge, MA: MIT Press.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artif. Intellig.* 112, 181–211. doi: 10.1016/S0004-3702(99)00052-1

Sutton, R. S., Rafols, E., and Koop, A. (2006). "Temporal abstraction in temporal-difference networks," in *Advances in Neural Information Processing Systems 18 (NIPS*05)* (Vancouver, BC).

Thomaz, A. L., and Breazeal, C. (2008). Experiments in socially guided exploration: Lessons learned in building robots that learn with and without human teachers. *Connect. Sci.* 20, 91–110. doi: 10.1080/09540090802091917

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

APPENDIX

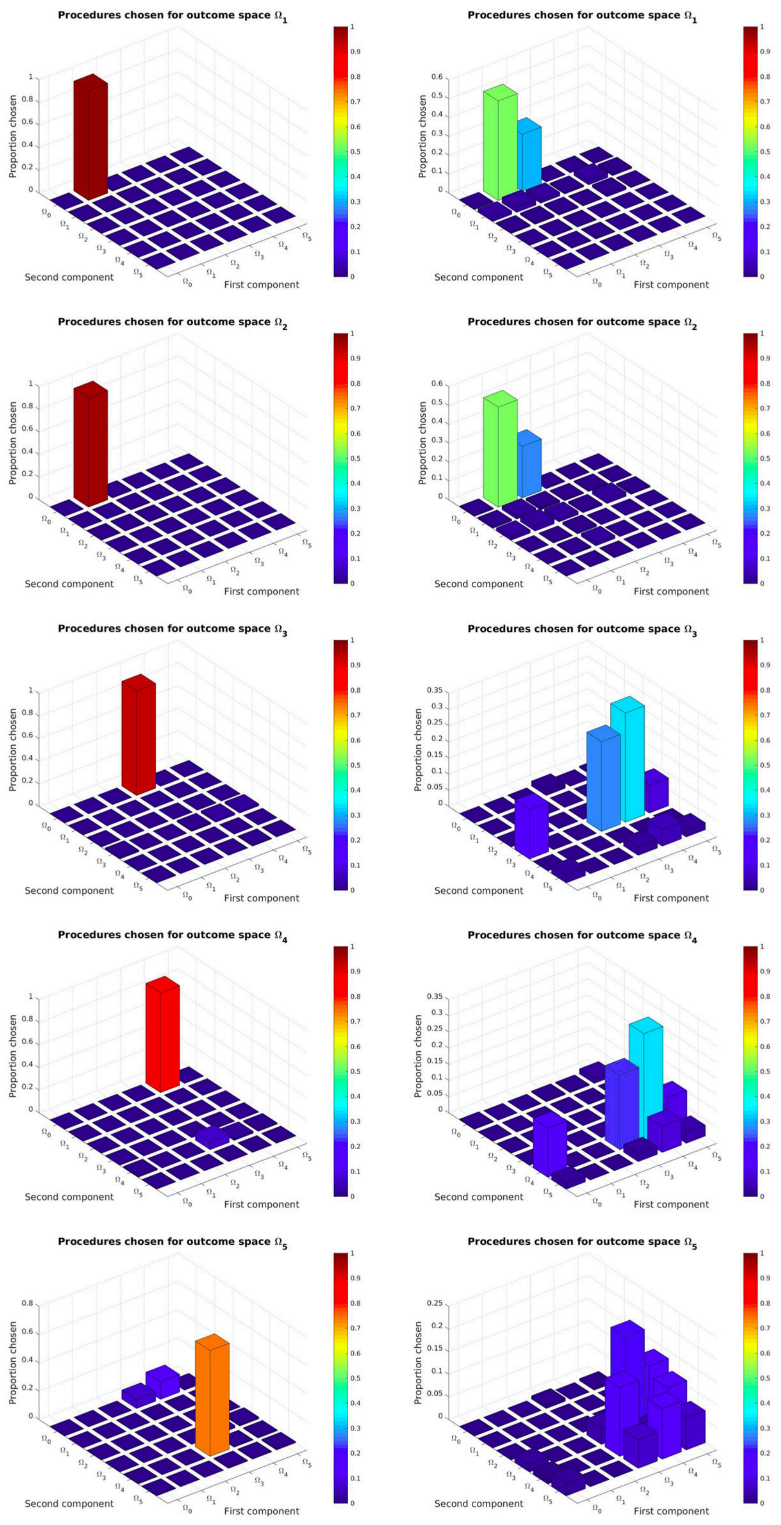


FIGURE A1 | Task hierarchy discovered by the SGIM-PB (left side) and IM-PB (right side) learners: this represents for each complex outcome space the percentage of time each procedural space would be chosen.

TABLE A1 | Student's *t*-test on two samples for comparing SGIM-PB with each of the algorithms and for comparing the procedure algorithms (SGIM-PB and IM-PB) to algorithms without the procedure framework (SGIM-ACTS, SAGG-RIAC and random).

		Global	Task0	Task 1	Task 2	Task 3	Task 4	Task 5
SGIM-PB vs. random	t	–33	9	–27	–15	–32	–50	–57
	p	3e-16	5e-8	9e-15	4e-11	4e-16	6e-19	5e-20
SGIM-PB vs. SAGG-RIAC	t	–3	9	–10	–2	–44	–46	–84
	p	1e-2	6e-8	1e-8	3e-2	4e-18	2e-18	1e-22
SGIM-PB vs. IM-PB	t	–11	–4	–4	–5	–5	–3	1
	p	3e-9	4e-4	1e-3	1e-4	9e-5	3e-3	0.2
SGIM-PB vs. SGIM-ACTS	t	–12	5	–3	–3	–0.5	–3	–18
	p	1e-9	2e-4	2e-3	1e-2	6e-2	1e-2	2e-12
(SGIM-PB, IM-PB) vs.	t	–2.5	9	–5	–2	–4	–5	–8
(random, SAGG-RIAC, SGIM-ACTS)	p	2e-2	1e-12	3e-6	7e-2	6e-4	3e-6	4e-11

We tested the difference of the distances to goal at the end of the learning ($t = 25,000$) for the global evaluation and for each task type. Negative values for t mean that SGIM-PB makes lower error. The non-significative results ($p > 0.1$) are highlighted.

frontiers in Neurorobotics

METHODS
published: 22 January 2019
doi: 10.3389/fnbot.2018.00088

Boredom-Driven Curious Learning by Homeo-Heterostatic Value Gradients

Yen Yu, Acer Y. C. Chang and Ryota Kanai*

Araya, Inc., Tokyo, Japan

This paper presents the Homeo-Heterostatic Value Gradients (HHVG) algorithm as a formal account on the constructive interplay between boredom and curiosity which gives rise to effective exploration and superior forward model learning. We offer an instrumental view of action selection, in which an action serves to disclose outcomes that have intrinsic meaningfulness to an agent itself. This motivated two central algorithmic ingredients: devaluation and devaluation progress, both underpin agent's cognition concerning intrinsically generated rewards. The two serve as an instantiation of homeostatic and heterostatic intrinsic motivation. A key insight from our algorithm is that the two seemingly opposite motivations can be reconciled—without which exploration and information-gathering cannot be effectively carried out. We supported this claim with empirical evidence, showing that boredom-enabled agents consistently outperformed other curious or explorative agent variants in model building benchmarks based on self-assisted experience accumulation.

Keywords: curiosity, boredom, goal-directedness, intrinsic motivation, outcome devaluation, satiety, homeostatic motivation, heterostatic motivation

OPEN ACCESS

Edited by:
Vieri Giuliano Santucci, Istituto di Scienze e Tecnologie della Cognizione (ISTC), Italy

Reviewed by:
Erhard Wieser, Technische Universität München, Germany
James Danckert, University of Waterloo, Canada

****Correspondence:***
Yen Yu
yen.yu@araya.org

Received: *29 April 2018*
Accepted: *19 December 2018*
Published: *22 January 2019*

Citation:
Yu Y, Chang AYC and Kanai R (2019) Boredom-Driven Curious Learning by Homeo-Heterostatic Value Gradients. Front. Neurorobot. 12:88. doi: 10.3389/fnbot.2018.00088

1. INTRODUCTION

In this study, we present an instrumental view of action selection, in which an action serves to disclose outcomes that have intrinsic meaningfulness—i.e., that hold epistemic values—to an agent itself. The implication of this statement is twofold: (1) for agents whose innate goal appeals to their own knowledge gain, the occurrence of curiosity rests upon the devaluation of known knowledge (and hence goal-directedness); (2) boredom—consequential to devaluation—and curiosity entail a mutually reinforcing cycle for such kind of (meaningful) disclosure to ensue.

Animal studies have shown that learning stimulus-response (S-R) associations through action-outcome reinforcement is but one facet of instrumental behavior. Internally, animals may build models that assign values to reappraise experienced outcomes. This expands the landscape of instrumental behavior to include the stimulus-outcome-response (S-O-R) learning system—or goal-directed learning (Balleine and Dickinson, 1998). Goal-directed behavior is known in both empirical and computational approaches to support adaptive and optimal action selection (Adams and Dickinson, 1981; Adams, 1982; Mannella et al., 2016). Central to such behavioral adaptiveness is devaluation. This means for a given action-outcome pair the associated reinforcing signal is no longer monotonic. Instead, an outcome value will change with reappraisals in accordance with an agent's internal goal.

One classic paradigm of devaluation manipulates an agent's level of satiation based on food accessibility, leading to altered behavioral patterns. In the context of epistemic disclosure, an analogy can be drawn between devaluation and the emergence of boredom, in which one's assimilation of knowledge reduces the value of similar knowledge in future encounters.

The relationship between boredom and outcome devaluation has a long history in psychological research. Empirical findings indicated that boredom is reportedly accompanied by negative affective experiences, suggesting that experienced outcomes are intrinsically evaluated and considered as less valuable (Perkins and Hill, 1985; Vodanovich et al., 1991; Fahlman et al., 2009; van Tilburg and Igou, 2012; Bench and Lench, 2013).

Psychophysiological studies also demonstrated that boredom plays an active role in eliciting information-seeking behaviors. Subjects showing higher levels of reported boredom are accompanied by increased autonomic arousal, such as heart rate and galvanic skin response. These findings are in line with our key notion that boredom intrinsically and actively drives learning behaviors (Berlyne, 1960; London et al., 1972; Harris, 2000). Note, however, that this notion is contested and a matter of unsettled debate (e.g., Eastwood et al., 2012; Fahlman et al., 2013; Merrifield and Danckert, 2014; Danckert et al., 2018). It is therefore worth pointing out that boredom may be accompanied by a low arousal state (Barmack, 1939; Geiwitz, 1966; Mikulas and Vodanovich, 1993; Pattyn et al., 2008; Vogel-Walcutt et al., 2012).

A finding by Larson (1990) invites the speculation that a task set may interact with boredom, thereby modifying a subject's behavioral pattern to follow either low or high arousal states. This means boredom may merely signal a state of disengagement. Whether an agent's cognitive resources can be freely allocated to re-engage another task inherently depends upon the existence of a prohibiting condition. Larson's (1990) participants, who reported boredom and were later rated with low scores in creative writing, were by design not allowed to disengage from the essay-writing task. Other theories, on the other hand, suggested that boredom is associated with increase in creativity (Schubert, 1977, 1978; Harris, 2000).

We thus postulate that, in the absence of any *a priori* cognitive or behavioral constraints, a state of boredom is followed by an attempt to diversify one's experience. That is, boredom begets exploration. This is in line with Vodanovich and Kass's (1990) notion of boredom in "inspiring a search for change and variety" and Zuckerman's (2008) "sensation-seeking." Sensation-seeking (Zuckerman, 1971, 2008; Kass and Vodanovich, 1990; Dahlen et al., 2005) is categorized as a personality trait, tightly linked to boredom susceptibility (Zuckerman et al., 1978). High sensation seekers get bored more easily, suggesting that individuals susceptible to boredom are predisposed to seek novel sensations. As a result, a learner who is also a novelty-seeker may have a world model that generalizes better. In our framework, receiving novel sensations is formalized as planning to visit states where an agent can effectively learn faster (i.e., the agent gets bored quicker). This effect is then treated as an intrinsic reward, prompting an agent to continue experiencing the state before the reward is depleted.

A recent computational modeling tapped into a similar theme (Gomez-Ramirez and Costa, 2017), where boredom facilitates exploration. However, our work differs from that of Gomez-Ramirez and Costa (2017) in that our model permits a simple form of agency (by having an action policy) and focuses on learning. Additionally, their exploration may favor predictable state space, whereas our agent will treat high predictability as an intrinsically non-rewarding state.

Finally, in psychology studies, the term boredom usually comes under two distinct constructs: a state of boredom and boredom proneness (Elpidorou, 2014, 2017; Mugon et al., 2018). Boredom proneness is regarded as the psychological predisposition of an individual to experience boredom which poses a systematic impact on one's social and psychological well-being. By contrast, a state of boredom is seen as a transient, regulatory signal that prompts one's behaviors into alignment with its goal-directedness (Elpidorou, 2017). In this sense, our model conceptually encompasses the function of the state boredom regulatory signal.

Curiosity, irrespective of being a by-product of external goal-attainment or an implicit goal in and of an agent itself, is often ascribed as a correlate of information-seeking behavior (Gottlieb et al., 2013). Behaviors exhibiting curious quality are observed in humans and animals alike, suggesting an universal role of curiosity in shaping one's fitness in terms of survival chance. Though the exact neural mechanism underlying the emergence of curious behavior still remains obscure, current paradigms have their focus on (1) novelty disclosure and (2) uncertainty reduction aspects of information-seeking (Bellemare et al., 2016; Friston et al., 2017; Ostrovski et al., 2017; Pathak et al., 2017). Indeed, both aspects can be argued to improve agent's fitness in epistemic landscape if the agent elects to incorporate the novelty or uncertainty.

Both boredom and curiosity are tightly connected to the notion of intrinsic motivation. Specifically, the occurrence of boredom and curiosity can be mapped to homeostatic and heterostatic motivations, respectively. The homeostatic and heterostatic motivations as two important classes of intrinsic motivation have been extensively reviewed in Oudeyer and Kaplan (2009). Simply, a homeostatic motivation drives a system to compensate perturbations in order to reach some equilibrial state. A heterostatic motivation is the opposite of a homeostatic motivation. A system that is driven by heterostatic motivations will self-perturb out of its equilibrium. In our formalism, predictive model learning and policy learning, each respectively induces boredom and curiosity, suggesting that the two classes of motivation can in fact be complementary when the two learning tasks are carried out concurrently. Our contribution thus pertains to the reconciliation of homeo-heterostatic motivations.

2. MARKOV DECISION PROCESS

In what follows, we briefly review preliminaries for the ensuing algorithm. We focus on well-established themes surrounding typical reinforcement learning, including Markov Decision Process and value gradients as a policy optimisation technique.

In Markov Decision Process (MDP) one considers the tuple $(S, A, R, P, \pi, \gamma)$. S and A are spaces of real vectors whose member, $\boldsymbol{s} \in S$ and $\boldsymbol{a} \in A$, represent states (or sensor values) and actions. R is some reward function defining the mapping $R : S \times A \rightarrow \mathbb{R}$. The probabilities associated with states and actions are given by the forward model $P(S'|A = \boldsymbol{a}, S = \boldsymbol{s})$ and the action policy $\pi(A|S = \boldsymbol{s})$. Throughout the paper we use the 'prime' notation, e.g., $\boldsymbol{s}'$, to represent one time step into the future: $\boldsymbol{s}' = \boldsymbol{s}(t+1)$.

The goal of MDP is to optimally determine the action policy π^* such that the expected cumulative reward over a finite (or infinite) horizon is maximized. Considering a finite horizon problem with discrete time, $t \in [0, T]$, this is equivalent to $\pi^* = \arg\max_\pi \mathbb{E}_{\boldsymbol{a}\sim\pi}\left[\sum_{t=0}^{T} \gamma^t R(\boldsymbol{s}(t), \boldsymbol{a}(t))\right]$, where $\gamma \in [0, 1]$ is the discount factor.

Many practical approaches for solving MDP often resort to approximating state-action value $q(\boldsymbol{a}, \boldsymbol{s})$ or state value $v(\boldsymbol{s})$ functions (Sutton and Barto, 1998; Mnih et al., 2013; Heess et al., 2015; Lillicrap et al., 2015). These value functions are given in the Bellman equation

$$\begin{aligned} v(\boldsymbol{s}) &= \mathbb{E}_{\pi(\boldsymbol{a}|\boldsymbol{s})}\Big[R(\boldsymbol{a}, \boldsymbol{s}) + \gamma q(\boldsymbol{a}, \boldsymbol{s})\Big] \\ &= \mathbb{E}_{\pi(\boldsymbol{a}|\boldsymbol{s})}\Big[R(\boldsymbol{a}, \boldsymbol{s}) + \gamma \mathbb{E}_{P(\boldsymbol{s}'|\boldsymbol{a},\boldsymbol{s})}[v(\boldsymbol{s}')]\Big] \qquad (1) \end{aligned}$$

When differentiable forward model and reward function are both available, policy gradients can be analytically estimated using value gradients (Fairbank and Alonso, 2012; Heess et al., 2015).

3. HOMEO-HETEROSTATIC VALUE GRADIENTS

This section describes formally the algorithmic structure and components of the Homeo-Heterostatic Value Gradients, or HHVG. The naming of HHVG suggests its connections with homeostatic and heterostatic intrinsic motivations. A detailed review on homeostatic and heterostatic motivations are given in Oudeyer and Kaplan (2009). Briefly, a homeostatic motivation encourages an organism to occupy a set of predictable, unsurprising states (i.e., a *comfort zone*). Whereas, a heterostatic motivation does the opposite; curiosity belongs to this category.

The algorithm offers a reconciliation between the two seemingly opposite qualities and concludes with their cooperative nature. Specifically, the knowledge an organism maintains about its comfort zone helps instigate outbound heterostatic drives. In return, satisfying heterostatic drives broadens the organism's extent of comfort zone. As a consequence, the organism not only improves its fitness in terms of homeostatic outreach but also becomes effectively curious.

3.1. Nomenclature and Notations

It is instructive to overview the nomenclature of the algorithm. We consistently associate homeostatic motivation with the emergence of *boredom*, which reflects the result of having incorporated novel information into one's knowledge, thereby diminishing the novelty to begin with. This is conceptually compatible with outcome *devaluation* or induced satiety in instrumental learning. *Devaluation progress* is therefore referred to as one's epistemic achievement. That is, the transitioning of a priori knowledge to one of having assimilated otherwise unknown information. The devaluation progress is interpreted as an instantiation of intrinsic reward. The drive to maintain steady rewards conforms to a heterostatic motivation.

The notation $\mathcal{L}(\cdot)$ consistently denotes loss functions throughout the paper; any variables on which the loss function

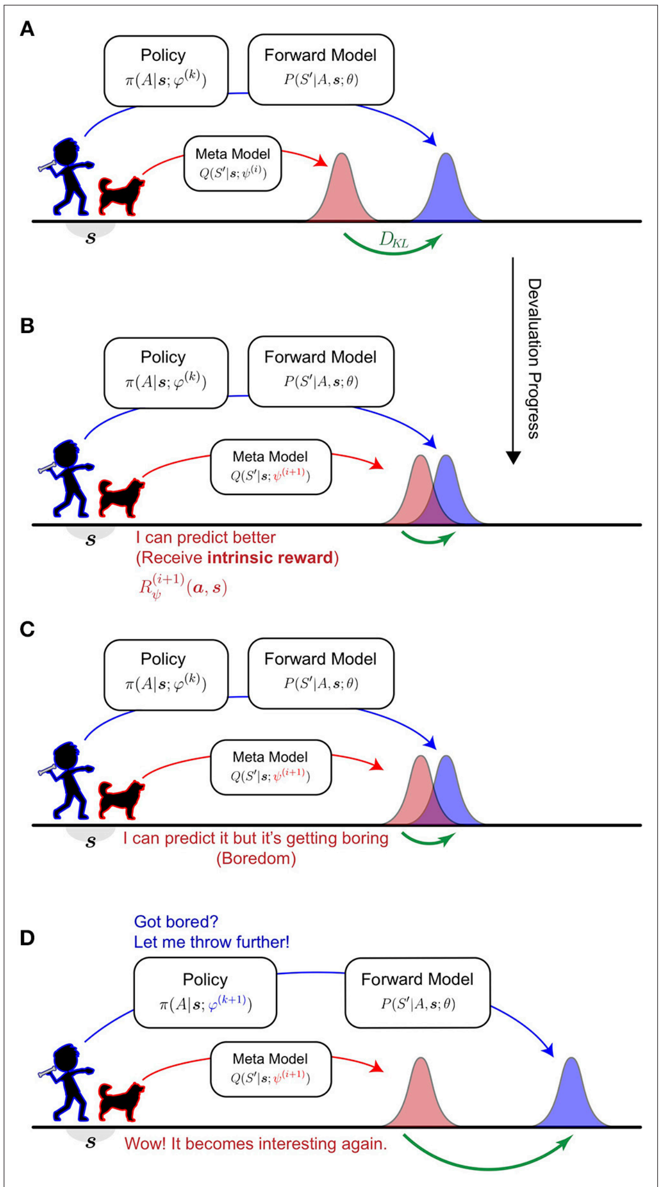

A
Policy
$\pi(A|\boldsymbol{s};\varphi^{(k)})$
Forward Model
$P(S'|A,\boldsymbol{s};\theta)$
Meta Model
$Q(S'|\boldsymbol{s};\psi^{(i)})$
$\boldsymbol{s}$
D_{KL}
Devaluation Progress
B
Meta Model
$Q(S'|\boldsymbol{s};\psi^{(i+1)})$
I can predict better
(Receive intrinsic reward)
$R_\psi^{(i+1)}(\boldsymbol{a},\boldsymbol{s})$
C
I can predict it but it's getting boring
(Boredom)
D
Got bored?
Let me throw further!
$\pi(A|\boldsymbol{s};\varphi^{(k+1)})$
Wow! It becomes interesting again.

FIGURE 1 | Intuitive understanding of the Homeo-Heterostatic Value Gradients (HHVG) algorithm. **(A)** The algorithm can be interpreted as the cooperative interplay between a thrower (kid; blue) and a catcher (dog; red). The thrower is equipped with a forward model that estimates its aiming and is controlled by an action policy. Without knowing the thrower's policy, the catcher (meta-model), in order to make good catches, infers where the thrower is aiming on average. **(B)** The catcher is interested in novel, unpredicted throws. Whenever the catcher improves its predictive power some intrinsic reward (devaluation progress) is generated. **(C)** As the catcher progresses further, similar throws become highly predictable, thus inducing a sense of boredom. **(D)** To make the interplay interesting again, the thrower is driven to devise new throws, so that the catcher can afford to make further progress. By repeating **(A,B)** the thrower has attempted diverse throws and known well about its aim. At the same time, the catcher will assume a vantage point for any throw.

depends are always made explicit. There are occasions where we abbreviated the loss function to avoid clutters. A definition such as $\mathcal{L}_{mm}(\psi) := \mathcal{L}(\boldsymbol{a}, \boldsymbol{s}; \psi, \theta)$ is then given upon first appearance. Here, the subscript *mm* indicates *meta-model*. One may tell in this example that the symbols $\boldsymbol{a}$, $\boldsymbol{s}$, and θ on the right hand side are temporarily omitted. This means the optimisation procedure for the meta-model concerns only the parameter ψ. Similarly, this applies to $\mathcal{L}_{fm}$, $\mathcal{L}_{vf}$, and $\mathcal{L}_{ap}$, where the subscripts stand for *forward model*, *value function*, and *action policy*. The symbol $\mathcal{N}$ is reserved for Normal distribution.

3.2. Intuition

An intuitive understanding of HHVG is visualized in **Figure 1**. Imagine the interplay between a thrower and their counterpart—a catcher. The catcher anticipates where the thrower is aiming and makes progress by improving its prediction. The thrower, on the other hand, keeps the catcher engaged by devising novel aims. Over time, the catcher knows well what the thrower is capable of, whilst the thrower has attempted a wide spectrum of pitches.

In the algorithm, the thrower is represented by a forward model attached to a controller (policy) and the catcher a "meta-model." We unpack and report them individually. Procedural information is summarized in **Algorithm 1**.

3.3. Forward Model

We start by specifying at current time the state and action sample as $\boldsymbol{s}$ and $\boldsymbol{a}$. The forward model describes the probability distribution over future state S', given $\boldsymbol{s}$, $\boldsymbol{a}$, and parameter θ.

$$P(S'|A = \boldsymbol{a}, S = \boldsymbol{s}; \theta) \tag{2}$$

The entropy associated with S', conditioned on $\boldsymbol{s}$ and $\boldsymbol{a}$, gives a measure of the degree to which S' is informative on average. We referred to this measure as one of *interestingness*. Note this is a different concept from the "interestingness" proposed by Schmidhuber (2008), which is the first-order derivative of compressibility.

3.4. Boredom, Outcome Devaluation, and Meta-Model

Boredom, in common understanding, is perhaps not unfamiliar to most people under the situation of being exposed to certain information which one has known well by heart. It is the opposite of being interested. In the current work, we limited the exposure of information to those being disclosed by one's actions.

To mark the necessity of boredom, we first identify the limitation of a naive instantiation of curiosity; then, we show that the introduction of boredom serves to resolve this limitation.

Consider the joint occurrence of future state S' and action A: $P(S', A|S = \boldsymbol{s}; \theta, \varphi)$. This can be derived from the product rule of probability using $P(S'|A = \boldsymbol{a}, S = \boldsymbol{s}; \theta)$ (as shown Equation 2) and action policy $\pi(A|S = \boldsymbol{s}; \varphi)$, parametrised by φ (action policy is revisited in section 3.6).

A naive approach to curiosity is by optimizing the action policy, such that A is predictive of maximum *interestingness* (see section 3.3) about the future.

However, this naive approach would certainly lead to the agent behaving habitually and, as a consequence, becoming obsessive about a limited set of outcomes. In other words, a purely interestingness-seeking agent is a darkroom agent (see section 3.7; also Friston et al., 2012 for related concept).

Algorithm 1 Homeo-heterostatic value gradients

```
1: Variables
        outer loop time $t$
        gradient step counter $\ell, i, j, k$
        state $\boldsymbol{s}^t := \boldsymbol{s}(t)$ and action $\boldsymbol{a}^t := \boldsymbol{a}(t)$
        learning rate $\lambda^\theta, \lambda^\psi, \lambda^\nu, \lambda^\varphi$
        discount factor $\gamma$
        experience pool $\mathcal{D}$
2: Models and parameters
        forward model $P(S'|\boldsymbol{s}, \boldsymbol{a}; \theta)$
        meta-model $Q(S'|\boldsymbol{s}; \psi)$
        value approximator $v(\boldsymbol{s}; \nu)$
        action policy $\pi(A|\boldsymbol{s}; \varphi)$
3: Objectives
        forward-model learning $\mathcal{L}_{fm}(\theta)$
        meta-model learning $\mathcal{L}_{mm}(\psi)$                  ▷ Eq.4
        value learning $\mathcal{L}_{vf}(\nu)$                        ▷ Eq.6
        policy learning $\mathcal{L}_{ap}(\varphi)$                   ▷ Eq.8
4: for $t = 0 \dots T$ do
5:     From $\boldsymbol{s}^t$, sample action $\boldsymbol{a}^t \sim \pi(\cdot|\boldsymbol{s}^t; \varphi)$
6:     Perform $\boldsymbol{a}^t$ and advance to $\boldsymbol{s}^{t+1}$
7:     Insert tuple $\left(\boldsymbol{s}^t, \boldsymbol{a}^t, \pi(\boldsymbol{a}^t|\boldsymbol{s}^t), \boldsymbol{s}^{t+1}\right)$ into $\mathcal{D}$
8:     Sample $\mathcal{D}$ and train forward model:
9:         $\mathcal{L}_{fm}(\theta) := \mathcal{L}(\boldsymbol{s}', \boldsymbol{a}, \boldsymbol{s}; \theta) = \|\boldsymbol{s}' - f(\boldsymbol{a}, \boldsymbol{s}; \theta)\|^2$   ▷ Eq.14
10:        $\theta^{(\ell+1)} \leftarrow \theta^{(\ell)} - \lambda_\theta \nabla_\theta \mathcal{L}_{fm}(\theta^{(\ell)})$
11:    Value learning ($M$ updates, see Algorithm 2)
12:    Sample $\mathcal{D}$ and perform devaluation:
13:        $\psi^{(i+1)} \leftarrow \psi^{(i)} - \lambda_\psi \nabla_\psi \mathcal{L}_{mm}(\psi^{(i)})$
14:    Sample $\mathcal{D}$ and train action policy:
15:        evaluate $R_\psi^{(i+1)} = \mathcal{L}_{mm}(\psi^{(i)}) - \mathcal{L}_{mm}(\psi^{(i+1)})$
16:        evaluate $v' = v(\boldsymbol{s}'; \nu^{(j+M)})$
17:        $w \leftarrow \pi(\boldsymbol{a}|\boldsymbol{s}; \varphi^{(k)}) / \pi(\boldsymbol{a}|\boldsymbol{s}; \varphi^{(<k)})$
18:        $\varphi^{(k+1)} \leftarrow \varphi^{(k)} + \lambda^\varphi \nabla_\varphi w \mathcal{L}_{ap}(\varphi^{(k)})$ given $R_\psi^{(i+1)}$, $v'$
```

Algorithm 2 Fitted Policy Evaluation [cf. Heess et al. (2015)]

```
1: Given
        outer loop time $t$
        experience pool $\mathcal{D}$
        value function $v(\boldsymbol{s}; \nu^{(j)})$
        gradient step counter $i, j, k$
2: Clone parameter $\tilde{\nu} \leftarrow \nu^{(j)}$
3: for $m = 1 \dots M$ do
4:     Sample $\left(\boldsymbol{s}^\tau, \boldsymbol{a}^\tau, \pi(\boldsymbol{a}^\tau|\boldsymbol{s}^\tau; \varphi^{(<k)}), \boldsymbol{s}^{\tau+1}\right)$ from $\mathcal{D}$ $(\tau < t)$
5:     Evaluate $R_\psi^{(i+1)} = \mathcal{L}_{mm}(\psi^{(i)}) - \mathcal{L}_{mm}(\psi^{(i+1)})$
6:     $y = R_\psi^{(i+1)} + \gamma v(\boldsymbol{s}^{\tau+1}; \tilde{\nu})$
7:     $w = \pi(\boldsymbol{a}^\tau|\boldsymbol{s}^\tau; \varphi^{(k)}) / \pi(\boldsymbol{a}^\tau|\boldsymbol{s}^\tau; \varphi^{(<k)})$
8:     Apply updates $\nu^{(j+m)} \leftarrow \nu^{(j+m-1)} - \nabla_\nu \frac{w}{2}\left(y - v(\boldsymbol{s}; \nu^{(j+m-1)})\right)^2$
9:     Every $C$ updates, $\tilde{\nu} \leftarrow \nu^{(j+m)}$
```

Such obsession with limited outcomes poses a caveat—the agent has no recourse to inform itself about prior exposure of similar sensations. If the agent is otherwise endowed with this capacity, namely, by assimilating previous experiences into summary statistics, an ensuing sense of boredom would be induced. The induction of boredom essentially causes the agent to value the same piece of information less, thus changing the agent's perception toward interestingness. If the agent were to pursue the same interestingness-seeking policy, a downstream effect of boredom would drive the agent to seek out other information that could have been known. This conception amounts to an implicit goal of *devaluating* known outcomes.

To this end, we introduce the following meta-model Q to represent *a priori* knowledge about the future. Note that Q is a conditional probability function over S' and is not to be confused with a state-action value function $q(\boldsymbol{a}, \boldsymbol{s})$ in MDP. The meta-model, parametrised by ψ, is an approximation to the *true* marginalization of joint probability $P(S', A|S = \boldsymbol{s}; \theta, \varphi)$ over A:

$$\begin{aligned} Q(S'|S = \boldsymbol{s}; \psi) &\approx P(S'|S = \boldsymbol{s}; \theta, \varphi) \\ &= \sum_A \Big[P(S', A|\boldsymbol{s}; \theta, \varphi) \Big] \\ &= \sum_A \Big[P(S'|A, \boldsymbol{s}; \theta) \pi(A|\boldsymbol{s}; \varphi) \Big] \end{aligned} \tag{3}$$

We associate the occurrence of boredom, or, synonymously, outcome devaluation, with minimizing the devaluation objective with respect to ψ. The devaluation objective is given by the Kullback-Leibler (KL) divergence:

$$\begin{aligned} \mathcal{L}_{mm}(\psi) &:= \mathcal{L}(\boldsymbol{a}, \boldsymbol{s}; \psi, \theta) \\ &= D_{KL} \left[P(\boldsymbol{s}'|\boldsymbol{a}, \boldsymbol{s}; \theta) \middle\| Q(\boldsymbol{s}'|\boldsymbol{s}; \psi) \right] \end{aligned} \tag{4}$$

3.5. Devaluation Progress, Intrinsic Reward, and Value Learning

Through the use of KL-divergence in Equation 4, we emphasize the complementary nature of devaluation in relation to a knowledge-gaining process. That is to say, devaluation results in information gain for the agent. This, in fact, can be regarded as cognitively rewarding and, thus, serves to motivate our definition of intrinsic reward.

One rewarding scenario happens when $Q(S'|\boldsymbol{s}; \psi)$ has all the information there is to be possessed by A about S'. A is therefore rendered redundant. One may speculate, at this point, the agent could opt for inhibiting its responses. Disengaging actions potentially saves energy which is rewarding in biological sense. This outcome is in line with the "opportunity cost model" proposed by Kurzban et al. (2013). In their model, boredom is seen as a resource regulatory signal which drives an agent to disengage the current task and curb the computational cost. As a consequence, the occurrence of boredom may encourage re-allocation of computational processes to alternative higher-value activities (Kurzban et al., 2013).

Alternatively, the agent may attempt to develop new behavioral repertoires, bringing into S' new information (i.e., novel outcomes) that is otherwise unknown to Q. The ensuing sections will focus on this line of thinking.

From Equation 4, we construct the quantity *devaluation progress* to represent an intrinsically motivated reward. The devaluation progress is given by the difference between KL-divergences before and after devaluation [as indicated by the superscript $(i + 1)$]:

$$\begin{aligned} R_{\psi}^{(i+1)}(\boldsymbol{a}, \boldsymbol{s}) &:= \mathcal{L}(\boldsymbol{a}, \boldsymbol{s}; \psi^{(i)}, \theta) - \mathcal{L}(\boldsymbol{a}, \boldsymbol{s}; \psi^{(i+1)}, \theta) \\ &= \mathcal{L}_{mm}(\psi^{(i)}) - \mathcal{L}_{mm}(\psi^{(i+1)}), \end{aligned} \tag{5}$$

Here, we write $R_{\psi}^{(i+1)}(\boldsymbol{a}, \boldsymbol{s})$ in accordance with notational convention in reinforcement learning, where reward is typically a function of state and action. Subscript ψ indicates the dependence of R on meta model parameter.

Having established the intrinsic reward, value learning is such that the value function approximator $v(\boldsymbol{s}; \nu)$ follows the Bellman equation $v(\boldsymbol{s}) = \mathbb{E}_{\boldsymbol{a}}[R(\boldsymbol{a}, \boldsymbol{s}) + \gamma \mathbb{E}_{\boldsymbol{s}'}[v(\boldsymbol{s}')]]$. In practice, we minimize the objective with respect to ν:

$$\begin{aligned} \mathcal{L}_{vf}(\nu) &:= \mathcal{L}(\boldsymbol{s}', \boldsymbol{a}, \boldsymbol{s}; \nu) \\ &= \left\| y - v(\boldsymbol{s}; \nu) \right\|^2 \\ y &= R_{\psi}^{(i+1)}(\boldsymbol{a}, \boldsymbol{s}) + \gamma v(\boldsymbol{s}'; \tilde{\nu}) \end{aligned} \tag{6}$$

3.6. Policy Optimisation

We define action policy at state $S = \boldsymbol{s}$ as the probability distribution over A with parameter φ:

$$\pi(A|S = \boldsymbol{s}; \varphi) \tag{7}$$

Our goal is to determine the policy parameter φ that maximizes the expected sum of future discounted rewards. One approach is by applying Stochastic Value Gradients (Heess et al., 2015) and maximizes the value function. We thus define our policy objective as follows (notice the negative sign; we used a gradient update rule that defaults to minimization):

$$\begin{aligned} \mathcal{L}_{ap}(\varphi) &:= \mathcal{L}(\boldsymbol{s}', \boldsymbol{a}, \boldsymbol{s}; \theta, \psi^{(i)}, \psi^{(i+1)}, \nu, \varphi) \\ &= -\mathbb{E}_{\boldsymbol{a} \sim \pi(\cdot|\boldsymbol{s}; \varphi)} \Big[R_{\psi}^{(i+1)}(\boldsymbol{a}, \boldsymbol{s}) + \gamma \mathbb{E}_{\boldsymbol{s}' \sim P(\cdot|\boldsymbol{a}, \boldsymbol{s}; \theta)} \big[v(\boldsymbol{s}'; \nu) \big] \Big] \end{aligned} \tag{8}$$

3.7. Remarks on Homeostatic and Heterostatic Regulations

Oudeyer and Kaplan (2009) outlined the distinctions between two important classes of intrinsic motivation: homeostatic and heterostatic. A homeostatic motivation is one that can be satiated, leading to a certain equilibrium behaviorally; whereas a heterostatic motivation topples the agent, thus preventing it from occupying habitual states.

Our algorithm entails regulations relating to both classes of intrinsic motivation. Specifically, the devaluation objective (Equation 4) realizes the homeostatic aspect due to its connection with induced satiety. On the other hand, the devaluation progress (Equation 5) introduced for policy optimisation instantiates a heterostatic drive to agent's behavioral pattern.

Heterostasis is motivated by the agent pushing itself toward novelty and away from devalued, homeostatic states (as revealed at the end of this section in Equation 13). This statement is shown formally by replacing the reward $R_{\psi}^{(i+1)}(\boldsymbol{a}, \boldsymbol{s})$ in Equation 8, with Equation 5. We then arrived at the following form involving expected KL-divergence:

$$\begin{aligned}
&- \mathbb{E}_{\boldsymbol{a}\sim\pi(\cdot|\boldsymbol{s};\varphi)}\Big[D_{KL}[P(\boldsymbol{s}'|\boldsymbol{a},\boldsymbol{s};\theta)\|Q(\boldsymbol{s}'|\boldsymbol{s};\psi^{(i)})] \\
&- D_{KL}[P(\boldsymbol{s}'|\boldsymbol{a},\boldsymbol{s};\theta)\|Q(\boldsymbol{s}'|\boldsymbol{s};\psi^{(i+1)})]\Big] \\
&- \mathbb{E}_{\boldsymbol{a}\sim\pi(\cdot|\boldsymbol{s};\varphi)}\mathbb{E}_{\boldsymbol{s}'\sim P(\cdot|\boldsymbol{a},\boldsymbol{s};\theta)}\Big[\nu(\boldsymbol{s}';\nu)\Big] \\
&= -\Big\{I(S':A|S=\boldsymbol{s};\psi^{(i)},\varphi,\theta) - I(S':A|S=\boldsymbol{s};\psi^{(i+1)},\varphi,\theta) \\
&+ \mathbb{E}_{\boldsymbol{a}\sim\pi(\cdot|\boldsymbol{s};\varphi)}\mathbb{E}_{\boldsymbol{s}'\sim P(\cdot|\boldsymbol{a},\boldsymbol{s};\theta)}\big[\nu(\boldsymbol{s}';\nu)\big]\Big\} \qquad (9)
\end{aligned}$$

Notice that the expected devaluation progress becomes the difference between conditional mutual information I before ($\psi^{(i)}$) and after devaluation ($\psi^{(i+1)}$).

Assume, for the moment, that the agent is equipped with devaluation capacity only. In other words, we replace the devaluation progress and fall back on devaluation objective, $R := \mathcal{L}_{mm}(\psi)$ (cf. Equation 5). The agent is now interestingness-seeking with homeostatic regulation. We further suppose that the dynamics of ψ and φ evolve in tandem, which gives

$$\begin{aligned}
I(S':A|S=\boldsymbol{s};\psi^{(i)},\varphi^{(k)}) &\to I(S':A|S=\boldsymbol{s};\psi^{(i+1)},\varphi^{(k)}) \\
&\to I(S':A|S=\boldsymbol{s};\psi^{(i+1)},\varphi^{(k+1)}) \\
&\to I(S':A|S=\boldsymbol{s};\psi^{(i+2)},\varphi^{(k+1)}) \to \dots \qquad (10)
\end{aligned}$$

In practice, the nature of devaluation and policy optimisation often depends on replaying agent's experience. Taking turn applying gradient updates to ψ and φ creates a self-reinforcing cycle that drives the policy to converge toward a point mass. For instance, if the policy is modeled by some Gaussian distribution, this updating scheme would result in infinite precision (zero spread).

For curiosity, however, such parameter dynamics should not be catastrophic if we subsume the homeostatic regulation and ensure the preservation of the relation given in Equation 11:

$$\begin{aligned}
I(S':A|S &= \boldsymbol{s};\psi^{(i+1)},\varphi^{(k)}) \le I(S':A|S=\boldsymbol{s};\psi^{(i)},\varphi^{(k)}) \\
&\le I(S':A|S=\boldsymbol{s};\psi^{(i+1)},\varphi^{(k+1)}) \\
\Rightarrow -I(S':A|S &= \boldsymbol{s};\psi^{(i+1)},\varphi^{(k)}) + I(S':A|S=\boldsymbol{s};\psi^{(i)},\varphi^{(k)}) \\
&\le I(S':A|S=\boldsymbol{s};\psi^{(i+1)},\varphi^{(k+1)}) \qquad (11)
\end{aligned}$$

This equation holds because the devaluation process on average has a tendency to make A less informative about S', after which A is perturbed to encourage a new S' less predictable to Q. By rearranging the equation such that the left hand side remains positive, we have arrived at a lower bound on $I(S':A|S = \boldsymbol{s};\psi^{(i+1)},\varphi^{(k+1)})$ which recovers the expected devaluation progress.

Equation 12 summarizes the argument associated with Equations (10, 11).

$$\begin{aligned}
\varphi^{(k+1)} &= \arg\max_{\varphi^{(k)}}\Big[I(S':A|S=\boldsymbol{s};\psi^{(i)},\varphi^{(k)}) \\
&\quad - \min_{\bar{\psi}^{(i)}} I(S':A|S=\boldsymbol{s};\bar{\psi}^{(i)},\varphi^{(k)})\Big] \\
&\ne \arg\max_{\varphi^{(k)}}\Big[\min_{\psi^{(i)}} I(S':A|S=\boldsymbol{s};\psi^{(i)},\varphi^{(k)})\Big] \qquad (12)
\end{aligned}$$

Finally, we offer an intuition on how policy optimisation gives rise to heterostatic motivation. This is made clear from the optimized target $I(S':A|S=\boldsymbol{s};\psi^{(i+1)},\varphi^{(k+1)})$, found on the right hand side of Equation 11. It is instructive to re-introduce the true marginalization $P(S'|S=\boldsymbol{s};\theta,\varphi)$ from Equation 3; write:

$$\begin{aligned}
&I(S':A|S=\boldsymbol{s};\psi^{(i+1)},\varphi^{(k+1)}) = \sum_{\boldsymbol{a}}\pi(\boldsymbol{a}|\boldsymbol{s};\varphi^{(k+1)}) \\
&\sum_{\boldsymbol{s}'}P(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a};\theta)\log\frac{P(\boldsymbol{s}'|\boldsymbol{a},\boldsymbol{s};\theta)}{Q(\boldsymbol{s}'|\boldsymbol{s};\psi^{(i+1)})} = \sum_{\boldsymbol{a}}\pi(\boldsymbol{a}|\boldsymbol{s};\varphi^{(k+1)}) \\
&\sum_{\boldsymbol{s}'}P(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a};\theta)\log\frac{P(\boldsymbol{s}'|\boldsymbol{a},\boldsymbol{s};\theta)}{P(\boldsymbol{s}'|\boldsymbol{s};\theta,\varphi^{(k+1)})}\frac{P(\boldsymbol{s}'|\boldsymbol{s};\theta,\varphi^{(k+1)})}{Q(\boldsymbol{s}'|\boldsymbol{s};\psi^{(i+1)})} \\
&= I(S':A|S=\boldsymbol{s};\varphi^{(k+1)}) + D_{KL}\big[P(\boldsymbol{s}'|\boldsymbol{s};\theta,\varphi^{(k+1)})\|Q(\boldsymbol{s}'|\boldsymbol{s};\psi^{(i+1)})\big] \qquad (13)
\end{aligned}$$

Simply, the optimized policy is such that the agent increases the conditional mutual information and is pushed away (via increasing the KL-divergence) from its homeostatic state Q.

4. IMPLEMENTATION CONSIDERATIONS

This section presents practical considerations when motivating the aforementioned agent using neural networks. These considerations were mainly for the ease of calculating KL-divergence analytically.

4.1. Forward Model

We assumed that the state follows some Gaussian distribution with mean $\boldsymbol{s}$ and covariance Σ. The future state is described by its mean $\boldsymbol{s}'$ according to the deterministic mapping $\boldsymbol{s}' = f(\boldsymbol{a},\boldsymbol{s};\theta)$, where $\boldsymbol{a}$ is the action sampled from policy. f represents a neural network with trainable parameter θ:

$$f(\boldsymbol{a},\boldsymbol{s};\theta) = \boldsymbol{A}\boldsymbol{s} + \left(\sum_{\iota} a_{\iota}\boldsymbol{B}^{\iota}\right)\boldsymbol{s} + \boldsymbol{C}\boldsymbol{a} + o \qquad (14)$$

$\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are approximations of Jacobian matrices and o a constant, all depending on θ. $\boldsymbol{B}$ is a three-way tensor indexed by ι along the first axis. This treatment is similar to Watter et al. (2015) (also cf. Karl et al., 2016), except that we considered a bilinear approximation and that, in the following sections, we used only the mean states in a deterministic environment.

The above formalism follows that $\boldsymbol{s}'$ has covariance matrix $\mathbb{E}[\boldsymbol{s}'\boldsymbol{s}'^{\intercal}] = \boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^{\intercal}$, where $\boldsymbol{J} = \left(\boldsymbol{A} + \sum_{\iota} a_{\iota}\boldsymbol{B}^{\iota}\right)$. The transition probability is then given by

$$P(S'|A = \boldsymbol{a}, S = \boldsymbol{s}; \theta) = \mathcal{N}\left(f(\boldsymbol{a}, \boldsymbol{s}; \theta), \boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^{\intercal}\right) \tag{15}$$

The model parameter θ represented four fully connected layers of width 512; the four layers were complemented by a residual connection, which was a single fully connected layer. We used rectified linear units (ReLU) as output nonlinearities. Next, four fully connected, linear layers each mapped the 512-dimensional output into vectors of dimension 16, 32, 8, and 1. These vectors were then reshaped into tensors and used as $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and o.

4.2. Meta Model

Our meta model was defined as a Gaussian distribution $Q(S'|S = \boldsymbol{s}; \psi) = \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\Sigma}'; \psi)$, where the mean $\boldsymbol{\mu}'$ and covariance matrix $\boldsymbol{\Sigma}'$ are outputs of a neural network parametrized by ψ. Specifically, to construct the covariance matrix, we used the fact that the eigendecomposition of a positive semi-definite matrix always exists. This then means we can use neural networks to specify an orthogonal matrix $\boldsymbol{H}$ and a diagonal matrix $\boldsymbol{D}$, such that the covariance matrix is equivalent to:

$$\begin{aligned} \boldsymbol{\Sigma}' &= \boldsymbol{H}\boldsymbol{D}\boldsymbol{H}^{\intercal}, \quad \boldsymbol{D} = \operatorname{diag}(\boldsymbol{d}) \\ \boldsymbol{H} &= \boldsymbol{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^{\intercal}}{\|\boldsymbol{u}\|^2}, \end{aligned} \tag{16}$$

where $\boldsymbol{d}$ is a positive-valued vector that specifies the diagonal elements of $\boldsymbol{D}$. The second line of Equation 16 shows how an orthogonal matrix can be built from a real-valued vector $\boldsymbol{u}$, called Householder vector (Tomczak and Welling, 2016). $\boldsymbol{I}$ is an identity matrix.

The network architecture used to compute μ', $\boldsymbol{d}$, and $\boldsymbol{u}$ consisted of three trainable layers, each of which was identically structured. Three fully connected layers with ReLU activation functions, complemented by a residual connection, were followed by a linear, fully connected output layer. The output layer for $\boldsymbol{d}$ used a Softplus nonlinearity to ensure positive values.

We can, of course, let the neural network output a full matrix $\boldsymbol{X}$ and have $\boldsymbol{\Sigma}' = \boldsymbol{X}\boldsymbol{X}^{\intercal}$. However, our method is less costly when scaling up the problem dimension.

4.3. Policy and Value Functions

Both the policy and value functions were identically structured in terms of network architecture. They consisted of four fully connected layers with ReLU activation functions, complemented by a residual connection. This was then followed by a linear output layer. The outputs for the policy network were treated as logits of a categorical distribution over action space.

5. EXPERIMENT

One testable hypothesis that emerges from our previous remark—that boredom gives rise to novelty seeking policy (cf. KL-divergence term in Equation 13)—is that *boredom helps improve agent's forward model learning*. This is because novelty seeking essentially implies diversity in agent's experience. In other words, a boredom-driven curious agent must exhibit a tendency toward *exploration* and against *perseveration*. This tendency is critical when the agent was not given a training set (on which it based its forward model learning) but has to self-assist in accumulating one from scratch.

Briefly, an agent that tends to explore would appear to accumulate experience that reflects a more complete picture of the environment and, therefore, leads to a more accurate forward model. By contrast, if an agent perseverates, it can only afford to occupy a limited set of states, leaving its forward model an inadequate representation of the environment.

The primary goal and purpose of the ensuing experiments is thus to illustrate, with and without boredom, (1) the extent to which an agent explores and perseverates, and (2) the forward model performance.

To this end, we motivated a model pruning hierarchy on which the comparisons above were based. The model pruning hierarchy, as summarized in **Table 1** and section 5.3, provides a principled way to assess agent's behavior by progressive degrading model components. As a result, the difference between a boredom agent and a boredom-free curious agent or non-curious agent can be highlighted.

Explorativeness and perseveration were assessed qualitatively using Coverage Rate (CR) and Coverage Entropy (CE), reported in section 6. CR simply counts the number of states an agent has visited amongst all possible states. CE focuses on weighing the number of time steps a state was being occupied. CR thus indicates the proportion of the environment explored by the agent. Whereas, a CE curve declining over time indicates the agent tends to perseverate around a limited state space.

Forward model performance was assessed based on validation error. The validation set was sampled from the oracle dataset (see section 5.2). Contrary to self-assisted data accumulation, the oracle dataset was acquired by uniformly sampling the state-action grid. This dataset is therefore an idealized case to learn the best possible forward model.

TABLE 1 | Model pruning hierarchy that helps highlight the contribution of boredom and curiosity in regulating agent's exploration and perseveration.

	Oracle	P/RW	PG/GR	PG/IRS	C/PE	C/B
FM	✓	✓	✓	✓	✓	✓
AP		○	✓	✓	✓	✓
IR				○	✓	✓
VF					✓	✓
MM						✓

Ticks mark the existence or dependence of trainable network components; circles indicate independent intervention. Top row: P/RW, random-walk policy; PG/GR, policy gradients with rewards drawn from a Gaussian distribution; PG/IRS, policy gradients with intrinsic reward samples; C/PE, curiosity using forward model error; C/B, curiosity from boredom. First column: FM, forward model; AP, action policy; IR, intrinsic rewards; VF, value function approximator; MM, meta-model.

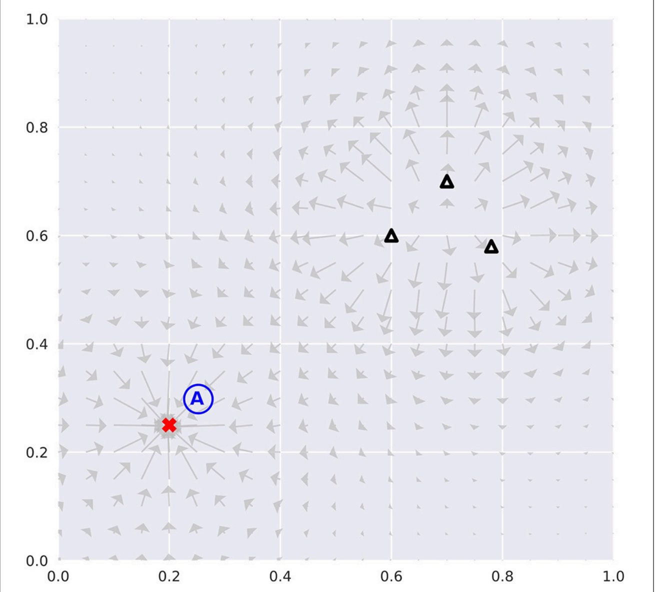


FIGURE 2 | Environmental configuration. The red cross represents an attractor, whilst black triangles repellers. Vector plots indicate the forces exerted if the agent assumed the positions with zero velocities. The initial position is set at the blue letter "A." This configuration remains identical cross all model variants and test runs.

Overall, we set the following constraints on training and environment conditions: (1) agent is responsible for assembling its own training set from scratch; (2) the probability of visiting different states is not uniformly distributed if the agent will commit to random walk; (3) the amount of time to accumulate training data points is limited.

5.1. Training Environment

Our agents were tested in a physics simulator, free of stochasticity, built to expand the classical Mountain Car environment (e.g., "MountainCar-v0" included in Brockman et al., 2016) into two-dimensional state space. The environment is analogous to the Mountain Car in ways that it has attractors and repellers that resemble hill- and valley-like landscapes (**Figure 2**). The presence of both structures serves as acceleration modifier to the agent. This makes state visitation biased toward attractors. Therefore, the acquisition of an accurate forward model necessitates planning visits to the vicinity of repellers.

The states an agent can occupy were defined as the tuple $(x, y, \dot{x}, \dot{y})$ in continuous real space. Positions $(x, y) \in [0, 1]^2$ were bounded in a unit square, whereas velocities $(\dot{x}, \dot{y})$ were not. Boundary condition resets x and y to zero velocities. However, it is possible for the agent to slide along the boundaries if its action goes in the direction parallel to the nearby boundary. We note that being trapped in the corners is possible; though an agent could potentially get itself unstuck if appropriate actions were carried out.

Agent's action policy was represented by a categorical distribution over accelerations in x and y directions. The distribution was defined on the interval $[-2.0, 2.0]^2$, evenly divided into a 11×11 grid. When an action is selected, the corresponding acceleration is modified according to forces exerted by the attractors and repellers.

Unlike the classical Mountain Car, our environment does not express external rewards, nor does it possess any states that are indicative of termination. Agents were allowed a pre-defined time limit ($T = 30,000$ steps; *Data Accumulation Phase* or DAP) to act without interruption. Agent's experiences in terms of state transitions were collected in a database, which was sampled from for training at each step. During DAP, learning rates for model parameters remained constant. After DAP (or *post*-DAP), agent entered an action-free stage lasted for $T = 30,000$, during which only sampling from own experience pool for forward model training was performed. Learning rate scheduling scheme was implemented at post-DAP.

An implementation of our training environment is available online [1].

5.2. Oracle Dataset

To contrast with self-assisted data accumulation, we constructed an oracle dataset. This dataset assumed unbiased state occupancy and action selection. We acquired the dataset by evenly dividing the state-action space into a $49 \times 49 \times 11 \times 11 \times 11 \times 11$ grid. Each state-action pair was passed to the physics simulator to evaluate the next state. The resultant tuple $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ then represents one entry in the dataset. The training, testing, and validation sets were prepared by re-sampling the resulting dataset without replacement according to the ratio 0.8, 0.16, and 0.04.

A class of model referred to as Oracle, which consists of a forward model only (**Table 1**), was trained on this dataset. The Oracle model does not need to learn an action policy, as actions are already specified in the oracle dataset. The Oracle model was trained for 60,000 epochs. During training, the learning rate was scheduled according to test error. Benchmarking was performed on the validation set as part of model comparisons (see section 5.4).

The oracle dataset differs from the ones that are populated by an agent as it explores. For instance, some locations in the state space are essentially inaccessible to our agent due to the force exerted by the repellers. These locations greatly inform forward model learning, however, but are only present in the oracle dataset and available to the Oracle model.

5.3. Model Pruning

We defined five variants of our boredom-driven curious agent. With each variation, the agent receives cumulative reductions in network components. Theses reductions are summarized as model pruning hierarchy in **Table 1**.

The reason that we motivated model comparisons based on model pruning is to emphasize the contribution of boredom and curiosity in regulating agent's explorativeness and perseveration. Overall, as model pruning progresses the agent was deprived of functional constructs like devaluation progress, intrinsic motivation, and planning. Eventually, the agent lost the ability to contextualize action selection and became a random-walk object. This corresponds to an ϵ-greedy policy with $\epsilon = 1$. A random-walk agent is explorative but it cannot be considered curious

[1]https://github.com/arayabrain/MountainCar2D

in the sense that no principled means are applied to regulate explorative behaviors. With the model variants detailed below we intended to demonstrate the impact boredom and intrinsic motivation have on regulating exploration and, as a consequence, on forward model learning.

5.3.1. Boredom-Driven Curiosity (C/B)

The first agent variant retained all distinctive components introduced in Section 3. The meta-model provides the devaluation progress as intrinsic rewards, whilst the value function enables the agent to plan actions that are intrinsically rewarding in the long run.

5.3.2. Predictive Error-Driven Curiosity (C/PE)

The C/PE variant tests whether the induction of boredom is a constructive form of intrinsic motivation. This is achieved by removing the meta-model, thereby requiring an alternative definition of intrinsic reward. We replaced the devaluation progress with *learning progress* defined by mean squared errors of the forward model:

$$\begin{aligned} R_{\theta}^{(\ell+1)} &:= \mathcal{L}_{fm}(\theta^{(\ell)}) - \mathcal{L}_{fm}(\theta^{(\ell+1)}) \\ \mathcal{L}_{fm}(\theta) &:= \mathcal{L}(\boldsymbol{s}', \boldsymbol{a}, \boldsymbol{s}; \theta) \\ &:= \|\boldsymbol{s}' - f(\boldsymbol{a}, \boldsymbol{s}; \theta)\|^2 \end{aligned} \tag{17}$$

The construction of learning progress is one typical approach to intrinsic motivation and curiosity (Schmidhuber, 1991; Pathak et al., 2017).

5.3.3. Policy Gradients, Intrinsic Reward Samples (PG/IRS), Gaussian Rewards (PG/GR)

Next, we examined how reward statistics alone influences policy update and, as a consequence, model learning. The value function was removed at this stage to dissociate policy learning from any downstream effects of value learning.

One distinctive feature of devaluation progress is that it entails time-varying rewards — depending on the amount of time over which an agent has evolved in the environment. We hypothesized that the emergence of curious policy is associated with reward dynamics over time. That is to say, if one perturbs the magnitudes and directions of the policy gradients with reward statistics appropriate for the ongoing time frame, the agent should exhibit similar curious behaviors. Nevertheless, we argue that such treatment is only sensible given virtually identical initial conditions. Specifically, all agent variants shared the same, environmental configuration, initial position, and network initialization.

To this end, we prepared a database for intrinsic reward samples. During C/B performance, all reward samples were collected and labeled with the corresponding time step. Afterwards, the PG/IRS agents randomly sampled from the database in a temporally synchronized manner and applied standard policy gradients.

The PG/IRS was contrasted with the PG/GR variant. Their difference lies in that a surrogate reward was used in place of the database. We defined the surrogate reward as a Gaussian distribution with time-invariant parameters, in which the mean $\mu = 0$ is under the assumption of equilibrium devaluation progress and the standard deviation $\sigma = 0.01$, as derived from the entire database.

5.3.4. Random-Walk Policy (P/RW)

Finally, we constructed a random-walk agent. All network components, apart from the forward model, were removed. This agent variant represents the case without intrinsic motivation and is agnostic to curiosity. Broadly speaking, the agent was still explorative due to its maximum entropy action policy. We regarded this version as the worse case scenario to contrast with the rest of the variants.

5.4. Model Comparisons

All model variants were compared on the basis of validation error given the oracle dataset. We performed 128 runs for each of the six variants (Oracle, C/B, C/PE, PG/IRS, PG/GR, and P/RW). All variants, across all runs, were assigned to identical environmental configuration (e.g., initial position, attractor/repeller placements). Network components, whenever applicable, shared identical architecture and were trained with consistent batch size and learning rate. Model parameters followed the Xavier initialization (Glorot and Bengio, 2010). During post-DAP, learning rate scheduling was implemented such that a factor 0.1 reduction was applied upon a 3000-epoch loss plateau.

6. RESULTS

In this section, we offered qualitative and quantitative assessment of agent's behavioral pattern and performance across different agent variants. As established previously, an agent's performance in modeling its own environment necessarily depends on both explorative and non-perseverative behaviors. The overall picture being delivered here is that the boredom-driven curious agent (abbrev. C/B) exhibited stronger tendency toward exploration (**Figures 3A,B**) and against perseveration (**Figures 3C,D**). In accordance with our prediction, the forward model performance was significantly better for the boredom agent, as compared with other curious or non-curious variants (**Figure 4** and **Tables 2**, **3**).

We first characterized individual agent variants' qualities of being i) explorative and ii) perseverative. Active exploration is one defining attribute of curiosity (Gottlieb et al., 2013), simply because it differentiates between uncertain and known situations, thus giving rise to effective information acquisition. This, however, should be complemented with suppressed perseveration; namely, to prevent oneself from being permanently or dynamically captured—i.e., by the corners or the attractor.

The two qualities can be distinguished, as shown in **Figure 3**, by respective measures of Coverage Rate (CR) and Coverage Entropy (CE). The two measures were computed by first turning the state space into a 50 × 50 grid, ignoring velocities. CR keeps track of whether or not a grid cell has been visited and, at each time step, corresponds to the proportion of visited grid cells. A CR curve increasing over time indicates that an agent would be exploring new grid cells.

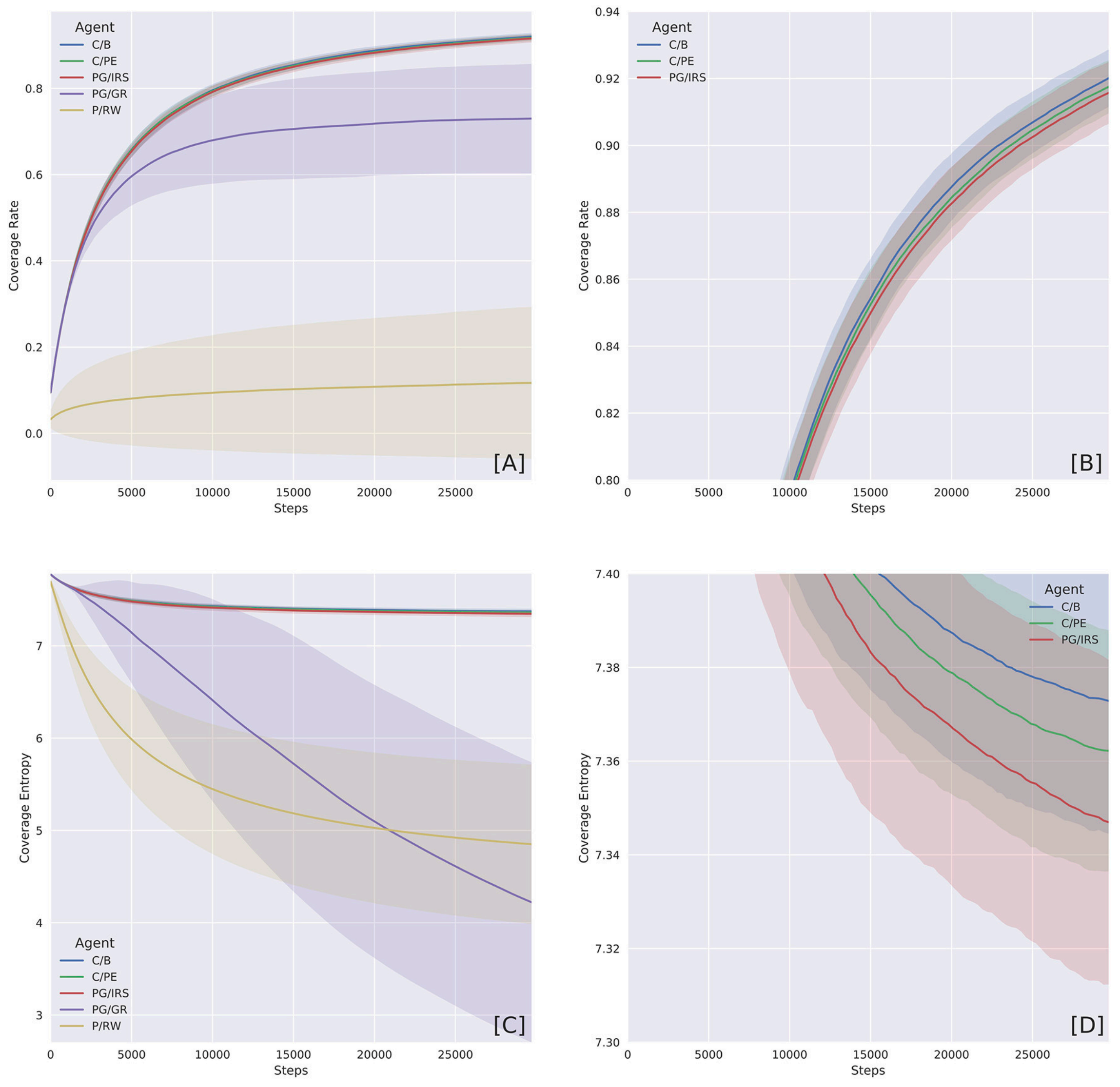


FIGURE 3 | Coverage Rate (CR) and Coverage Entropy (CE) by agent variants. The two measures were computed by first turning the state space into a 50 × 50 grid, ignoring velocities. CR then marks over time whether or not a cell has been visited. Whereas, CE treats the grid as a probability distribution. Starting with maximum entropy, CR cumulatively counts the number of times a position is being visited. Entropy was calculated at each time step using the normalized counter. **(A)** Overview of CR shows the distinction between curious and non-curious agents. Curiosity caused the agents to explore faster. **(B)** Close-up on the curious agent variants, which were equally explorative. **(C)** Overview of CE shows agents with different levels of perseverance. The P/RW variants were captured by the attractor, whilst the PG/GR variants were prone to blockage. **(D)** Close-up on curious agents, which were characterized by higher CE due to attractor avoidance and more frequent repeller visitation attempts. Shaded regions represent one standard deviation.

CE, on the other hand, accounts for the number of time steps an agent revisited one grid cell. This then gives an empirical probability distribution at each time step that reports the likelihood of finding an agent occupying a grid cell. A concentrated probability distribution means an agent only paid visit to a small set of grid cells and, as a result, the probability distribution has low entropy.

Because (state) visitation bias was inherent in our training environment, naturally, agents occupying a subset of states would cause CE to reduce faster than those who attempted to escape. The C/B, C/PE, and PG/IRS variants were regarded as curious and intrinsically motivated. Our results showed that these variants were predominantly explorative and non-perseverative. By contrast, the P/RW agent, albeit explorative, had no principled means to escape the potential well. However, if $t \rightarrow \infty$ the P/RW should be able to explore further by chance. The PG/GR variant, on the other hand, exhibited, intermediate explorativeness and extreme perseverance with disproportionately high variance. We attributed this behavior to the detrimental effects of inappropriately informative reward statistics.

Next, we benchmarked forward model performance of individual variants by their validation loss and error percentage. We reported DAP and post-DAP performances separately as a

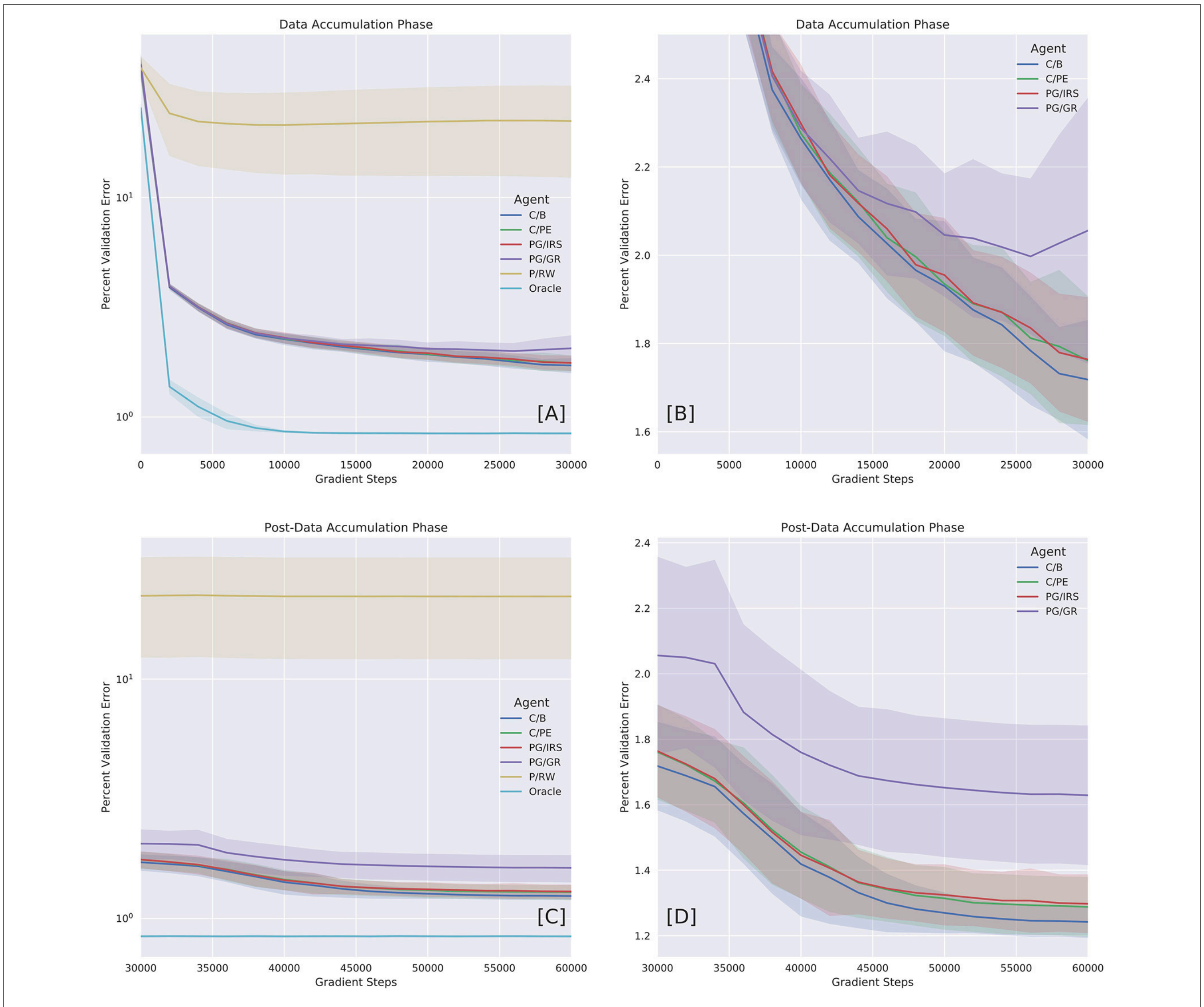


FIGURE 4 | Benchmarking model variants with oracle dataset. Performances were reported in error percentage (also, see **Table 2**). **(A)** Performance as a function of time during Data Accumulation Phase (DAP). **(B)** Close-up on curious variants (C/B, C/PE, and PG/IRS), as well as policy gradients (PG/GR) informed by surrogate reward statistics. The C/PE and PG/IRS variants performed similarly, but differed significantly from C/B **(Table 2)**. **(C)** Performance over time during post-DAP. **(D)** Close-up on post-DAP performances for curious variants and PG/GR.

function of time in **Figure 4**. Error percentage was calculated as the percent ratio between root mean squared loss and the maximum pair-wise Euclidean distance in the validation set. This ratio can be summarized by $\|\boldsymbol{s}'_k - f(\boldsymbol{a}_k, \boldsymbol{s}_k; \theta)\| / \max_{i,j} \|\mathcal{D}_i - \mathcal{D}_j\|$, where $\mathcal{D}$ is the validation set and $(\boldsymbol{s}'_k, \boldsymbol{a}_k, \boldsymbol{s}_k) \in \mathcal{D}$.

The Oracle model, trained under the supervision of oracle training set, reached an error percentage of 0.84% for both DAP and post-DAP, amounting to approximately 30% improvement over the terminal performance of the C/B variant. All variants considered curious (C/B, C/PE, and PG/IRS) had similar performances during DAP. In particular, the PG/IRS, which received independent intervention from the 'true' reward distributions achieved marginally lower performance but indistinguishable from the C/PE variant. This outcome was observed for both DAP and post-DAP, suggesting intrinsic reward samples derived from C/B contributed favorably even to the standard policy gradients algorithm.

Though without the ability to approximate value function, the PG/IRS variant underperformed in benchmarking, as compared with the value-enabled, C/B variant. Using non-parametric test, the difference was detected for DAP ($p = 0.0006$) and post-DAP ($p = 6.4\text{E-}8$), respectively. Similar observations were also made for comparisons between C/B and C/PE, at $p = 0.0029$ (DAP) and $p = 5.9\text{E-}5$ (post-DAP). Overall, this suggested significant differences in the experiences accumulated across agent variants. The aforementioned statistics were reported in **Tables 2**, **3**.

TABLE 2 | Summary statistics on validation loss and error percentage as benchmarking scores.

Agent	DAP		Post-DAP	
	MSE loss (SD)	Mean Percent Error (SD)	MSE loss (SD)	Mean Percent Error (SD)
Oracle	0.0008 (2.3E-5)	0.8430 (0.0123)	0.0008 (2.2E-5)	0.8428 (0.0114)
C/B	0.0033 (0.0006)	1.7181 (0.1357)	0.0017 (0.0001)	1.2420 (0.0488)
C/PE	0.0035 (0.0006)	1.7611 (0.1464)	0.0019 (0.0003)	1.2882 (0.0916)
PG/IRS	0.0035 (0.0006)	1.7637 (0.1418)	0.0020 (0.0003)	1.2976 (0.0902)
PG/GR	0.0048 (0.0017)	2.0559 (0.3026)	0.0030 (0.0008)	1.6288 (0.2140)
P/RW	0.6663 (0.3904)	22.2734 (10.0085)	0.6615 (0.3864)	22.1453 (10.0775)

Apart from the Oracle model, a trend of declining scores can be observed as the agent degraded from C/B to P/RW, indicating the contribution of boredom and curiosity in model learning. Key: DAP, Data Accumulation Phase; SD, standard deviation. For agent codes, see ***Table 1****.*

TABLE 3 | Non-parametric statistical tests comparing terminal performance at DAP and post-DAP for curious model variants.

Mann-Whitney U-Test ($n = 128, \alpha = 0.025$, Bonferroni corrected)			
Validation loss		DAP ($T = 30,000$)	Post-DAP ($T = 60,000$)
C/B < C/PE	Statistics	6558.0	5911.0
	p-value	0.0029	5.9E-5
C/B < PG/IRS	Statistics	6275.0	5062.0
	p-value	0.0006	6.4E-8

Following ***Table 2****, even though the boredom score came close to other curious variants (C/PE and PG/IRS), the boredom variant still outperformed the other two on statistical grounds.*

7. LIMITATION

One obvious limitation of the proposed method is scalability. We imposed Gaussian assumption on the forward model and meta-model because this lends the KL-divergence between the two to have a closed form solution. However, this solution depends on both matrix inversion and log-determinant, whose computational complexity normally falls around an order of 3 when using Cholesky decomposition. To circumvent this limitation, the intrinsic reward (devaluation progress) may be replaced with one based on (forward model) prediction error at the expense of lesser curiosity.

The Gaussian assumption also puts limitations on the expressiveness of the models. This can be slightly relaxed to admit Gaussian mixture models. KL-divergence between Gaussian mixture models is not tractable but can nonetheless be approximated (e.g., Hershey and Olsen, 2007). Alternatively, employing normalizing flows (Rezende and Mohamed, 2015) also allows expressive models. Calculating KL-divergence in this case is typically resorted to Monte Carlo approximation. These are potential extensions that can be applied to the current work in the future.

8. CONCLUSION

We have provided a formal account on the emergence of boredom from an information-seeking perspective and addressed its constructive role in enabling curious behaviors. Boredom thus motivates an instrumental view of action selection, in which an action serves to disclose outcomes that have intrinsic meaningfulness to an agent itself. This is, a bored agent must seek out information worth assimilating into itself. This led to the central claim of this study—pertaining to the superior data-gathering efficiency and hence effective curiosity. We supported this claim with empirical evidence, showing that boredom-enabled agents consistently outperformed other curious agents in self-assisted forward model learning. Our results solicited the interpretation that the relationship between homeostatic and heterostatic intrinsic motivations can in fact be complementary; therefore, we have offered one unifying perspective for the intrinsic motivation landscape.

Our proposed method is general in formalization and sits comfortably with existing MDP problems. Our future work is then to apply the method to more complex problems, such as embedding into a robot for real-world scenarios.

AUTHOR CONTRIBUTIONS

YY conceived of this study, performed the experiments, and wrote the first draft of the manuscript. AC programmed the physics simulator, wrote part of Introduction, and created **Figure 1**. All authors contributed to manuscript revision, read and approved the submitted version.

FUNDING

This study was funded by the Japan Science and Technology Agency (JST) under CREST grant number JPMJCR15E2.

ACKNOWLEDGMENTS

YY would like to thank Martin Biehl and Ildefons Magrans de Abril for insightful discussions.

REFERENCES

Adams, C. D. (1982). Variations in the sensitivity of instrumental responding to reinforcer devaluation. *Q. J. Exp. Psychol.* 34, 77–98. doi: 10.1080/14640748208400878

Adams, C. D., and Dickinson, A. (1981). Instrumental responding following reinforcer devaluation. *Q. J. Exp. Psychol. B* 33, 109–121. doi: 10.1080/14640748108400816

Balleine, B. W., and Dickinson, A. (1998). Goal-directed instrumental action: contingency and incentive learning and their cortical substrates.

Neuropharmacology 37, 407–419. doi: 10.1016/S0028-3908(98)00033-1

Barmack, J. E. (1939). A definition of boredom: a reply to Mr. Berman. *Am. J. Psychol.* 52, 467–471.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). "Unifying count-based exploration and intrinsic motivation," in *Advances in Neural Information Processing Systems*, eds D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (New York, NY: Curran Associates, Inc.), 1471–1479. doi: 10.3390/bs3030459

Bench, S. W., and Lench, H. C. (2013). On the function of boredom. *Behav. Sci.* 3, 459–472.

Berlyne, D. E. (1960). *Conflict, Arousal, and Curiosity*. New York, NY: McGraw-Hill Book Company. doi: 10.1037/11164-000

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym. *arXiv [Preprint]. arXiv:1606.01540*.

Dahlen, E. R., Martin, R. C., Ragan, K., and Kuhlman, M. M. (2005). Driving anger, sensation seeking, impulsiveness, and boredom proneness in the prediction of unsafe driving. *Accid. Anal. Prev.* 37, 341–348. doi: 10.1016/j.aap.2004.10.006

Danckert, J., Hammerschmidt, T., Marty-Dugas, J., and Smilek, D. (2018). Boredom: Under-aroused and restless. *Consciousness Cogn.* 61, 24–37. doi: 10.1016/j.concog.2018.03.014

Eastwood, J. D., Frischen, A., Fenske, M. J., and Smilek, D. (2012). The unengaged mind: defining boredom in terms of attention. *Perspect. Psychol. Sci.* 7, 482–495. doi: 10.1177/1745691612456044

Elpidorou, A. (2014). The bright side of boredom. *Front. Psychol.* 5:1245. doi: 10.3389/fpsyg.2014.01245

Elpidorou, A. (2017). The bored mind is a guiding mind: toward a regulatory theory of boredom. *Phenomenol. Cogn. Sci.* 35, 17. doi: 10.1007/s11097-017-9515-1

Fahlman, S. A., Mercer, K. B., Gaskovski, P., Eastwood, A. E., and Eastwood, J. D. (2009). Does a lack of life meaning cause boredom? results from psychometric, longitudinal, and experimental analyses. *J. Soc. Clin. Psychol.* 28, 307–340. doi: 10.1521/jscp.2009.28.3.307

Fahlman, S. A., Mercer-Lynn, K. B., Flora, D. B., and Eastwood, J. D. (2013). Development and validation of the multidimensional state boredom scale. *Assessment* 20, 68–85. doi: 10.1177/1073191111421303

Fairbank, M., and Alonso, E. (2012). "Value-gradient learning," in *Neural Networks (IJCNN), The 2012 International Joint Conference on IEEE* (Brisbane, QLD), 1–8.

Friston, K., Thornton, C., and Clark, A. (2012). Free-energy minimization and the dark-room problem. *Front. Psychol.* 3:130. doi: 10.3389/fpsyg.2012.00130

Friston, K. J., Lin, M., Frith, C. D., Pezzulo, G., Hobson, J. A., and Ondobaka, S. (2017). Active inference, curiosity and insight. *Neural Comput.* 29, 2633–2683. doi: 10.1162/neco_a_00999

Geiwitz, P. J. (1966). Structure of boredom. *Jo. Personal. Soc. Psychol.* 3, 592. doi: 10.1037/h0023202

Glorot, X., and Bengio, Y. (2010). "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Sardinia), 249–256.

Gomez-Ramirez, J., and Costa, T. (2017). Boredom begets creativity: a solution to the exploitation exploration trade-off in predictive coding. *BioSystems* 162, 168–176. doi: 10.1016/j.biosystems.2017.04.006

Gottlieb, J., Oudeyer, P.-Y., Lopes, M., and Baranes, A. (2013). Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends Cogn. Sci.* 17, 585–593. doi: 10.1016/j.tics.2013.09.001

Harris, M. B. (2000). Correlates and characteristics of boredom proneness and boredom. *J. Appl. Soc. Psychol.* 30, 576–598. doi: 10.1111/j.1559-1816.2000.tb02497.x

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Tassa, Y., and Erez, T. (2015). Learning continuous control policies by stochastic value gradients. *arXiv [Preprint]. arXiv:1510.09142*.

Hershey, J. R., and Olsen, P. A. (2007). "Approximating the kullback leibler divergence between gaussian mixture models," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on IEEE*, Vol. 4 (Honolulu, HI), IV–317.

Karl, M., Soelch, M., Bayer, J., and van der Smagt, P. (2016). Deep variational bayes filters: unsupervised learning of state space models from raw data. *arXiv [Preprint]. arXiv:1605.06432*.

Kass, S. J., and Vodanovich, S. J. (1990). Boredom proneness: its relationship to type a behavior pattern and sensation seeking. *Psychology* 27, 7–16.

Kurzban, R., Duckworth, A., Kable, J. W., and Myers, J. (2013). Cost-benefit models as the next, best option for understanding subjective effort. *Behav. Brain Sci.* 36, 707–726. doi: 10.1017/S0140525X13001532

Larson, R. W. (1990). Emotions and the creative process; anxiety, boredom, and enjoyment as predictors of creative writing. *Imagination Cogn. Personal.* 9, 275–292. doi: 10.2190/XT9G-WXRF-BK4M-36AK

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. *arXiv [Preprint]. arXiv:1509.02971*.

London, H., Schubert, D. S., and Washburn, D. (1972). Increase of autonomic arousal by boredom. *J. Abnorm. Psychol.* 80, 29. doi: 10.1037/h0033311

Mannella, F., Mirolli, M., and Baldassarre, G. (2016). Goal-directed behavior and instrumental devaluation: a neural system-level computational model. *Front. Behav. Neurosci.* 10:181. doi: 10.3389/fnbeh.2016.00181

Merrifield, C., and Danckert, J. (2014). Characterizing the psychophysiological signature of boredom. *Exp. Brain Res.* 232, 481–491. doi: 10.1007/s00221-013-3755-2

Mikulas, W. L., and Vodanovich, S. J. (1993). The essence of boredom. *Psychol. Rec.* 43, 3.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. *arXiv [Preprint]. arXiv:1312.5602*.

Mugon, J., Struk, A., and Danckert, J. (2018). A failure to launch: regulatory modes and boredom proneness. *Front. Psychol.* 9:1126. doi: 10.3389/fpsyg.2018.01126

Ostrovski, G., Bellemare, M. G., Oord, A. v. d., and Munos, R. (2017). Count-based exploration with neural density models. *arXiv [Preprint]. arXiv:1703.01310*.

Oudeyer, P.-Y., and Kaplan, F. (2009). What is intrinsic motivation? a typology of computational approaches. *Front. Neurorob.* 1:6. doi: 10.3389/neuro.12.006.2007

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). "Curiosity-driven exploration by self-supervised prediction," in *ICML* (Sydney, NSW).

Pattyn, N., Neyt, X., Henderickx, D., and Soetens, E. (2008). Psychophysiological investigation of vigilance decrement: boredom or cognitive fatigue? *Physiol. Behav.* 93, 369–378. doi: 10.1016/j.physbeh.2007.09.016

Perkins, R. E., and Hill, A. (1985). Cognitive and affective aspects of boredom. *Br. J. Psychol.* 76, 221–234. doi: 10.1111/j.2044-8295.1985.tb01946.x

Rezende, D. J., and Mohamed, S. (2015). Variational inference with normalizing flows. *arXiv [Preprint]. arXiv:1505.05770*.

Schmidhuber, J. (1991). "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats* (Paris), 222–227.

Schmidhuber, J. (2008). "Driven by compression progress: a simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes," in *Workshop on Anticipatory Behavior in Adaptive Learning Systems* (Munich: Springer), 48–76.

Schubert, D. S. (1977). Boredom as an antagonist of creativity. *J. Creat. Behav.* 11, 233–240. doi: 10.1002/j.2162-6057.1977.tb00631.x

Schubert, D. S. (1978). Creativity and coping with boredom. *Psychiatr. Ann.* 8, 46–54.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, Vol. 1. Cambridge, MA: MIT Press.

Tomczak, J. M., and Welling, M. (2016). Improving variational auto-encoders using householder flow. *arXiv [Preprint]. arXiv:1611.09630*.

van Tilburg, W. A., and Igou, E. R. (2012). On boredom: Lack of challenge and meaning as distinct boredom experiences. *Motiv. Emotion* 36, 181–194. doi: 10.1007/s11031-011-9234-9

Vodanovich, S. J., and Kass, S. J. (1990). A factor analytic study of the boredom proneness scale. *J. Personal. Asses.* 55, 115–123. doi: 10.1080/00223891.1990.9674051

Vodanovich, S. J., Verner, K. M., and Gilbride, T. V. (1991). Boredom proneness: Its relationship to positive and negative affect. *Psychol. Rep.* 69, 1139–1146. doi: 10.2466/pr0.1991.69.3f.1139

Vogel-Walcutt, J. J., Fiorella, L., Carper, T., and Schatz, S. (2012). The definition, assessment, and mitigation of state boredom within educational settings: a comprehensive review. *Educ. Psychol. Rev.* 24, 89–111. doi: 10.1007/s10648-011-9182-7

Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. (2015). Embed to control: a locally linear latent dynamics model for control from raw images. *arXiv [Preprint]. arXiv:1506.07365.*

Zuckerman, M. (1971). Dimensions of sensation seeking. *J. Consult. Clin. Psychol.* 36, 45–52. doi: 10.1037/h0030478

Zuckerman, M. (2008). "Sensation seeking," in *The International Encyclopedia of Communication, 1st Edn*, ed W. Donsbach (John Wiley & Sons, Ltd), 1–3. doi: 10.1002/9781405186407.wbiecs029

Zuckerman, M., Eysenck, S. B., and Eysenck, H. J. (1978). Sensation seeking in england and america: cross-cultural, age, and sex comparisons. *J. Consult. Clin. Psychol.* 46, 139. doi: 10.1037/0022-006X.46.1.139

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH
published: 29 January 2019
doi: 10.3389/frobt.2019.00002

Ostensive-Cue Sensitive Learning and Exclusive Evaluation of Policies: A Solution for Measuring Contingency of Experiences for Social Developmental Robot

Hamed Mahzoon, Yuichiro Yoshikawa and Hiroshi Ishiguro*

Intelligent Robotics Laboratory, Department of System Innovation, Graduate School of Engineering Science, Osaka University, Osaka, Japan

OPEN ACCESS

Edited by:
Vieri Giuliano Santucci, Istituto di Scienze e Tecnologie della Cognizione, Italy

Reviewed by:
Megan K. Strait, The University of Texas Rio Grande Valley, United States
Masaki Ogino, Kansai University, Japan

***Correspondence:**
Hamed Mahzoon
hamed.mahzoon@irl.sys.es.osaka-u.ac.jp

Specialty section:
This article was submitted to Humanoid Robotics, a section of the journal Frontiers in Robotics and AI

Received: *07 May 2018*
Accepted: *11 January 2019*
Published: *29 January 2019*

Citation:
Mahzoon H, Yoshikawa Y and Ishiguro H (2019) Ostensive-Cue Sensitive Learning and Exclusive Evaluation of Policies: A Solution for Measuring Contingency of Experiences for Social Developmental Robot. Front. Robot. AI 6:2. doi: 10.3389/frobt.2019.00002

Joint attention related behaviors (JARBs) are some of the most important and basic cognitive functions for establishing successful communication in human interaction. It is learned gradually during the infant's developmental process, and enables the infant to purposefully improve his/her interaction with the others. To adopt such a developmental process for building an adaptive and social robot, previous studies proposed several contingency evaluation methods, by which an infant robot becomes able to sequentially learn some primary social skills. These skills included gaze following and social referencing, and could be acquired through interacting with a human caregiver model in a computer simulation. However, to implement such methods to a real-world robot, two major problems, that were not addressed in the previous research, have remained unresearched: (1) dependency of histogram of the observed events by the robot to each other, which increases the error of the internal calculation and consequently decreases the accuracy of contingency evaluation; and (2) unsynchronized teaching/learning phase of the teaching-caregiver and the learning-robot, which leads the robot and the caregiver not to understand the suitable timing for the learning and the teaching, respectively. In this paper, we address these two problems, and propose two algorithms in order to solve them: (1) exclusive evaluation of policies (XEP) for the former, and (2) ostensive-cue sensitive learning (OsL) for the latter. To show the effect of the proposed algorithms, we conducted a real-world human-robot interaction experiment with 48 subjects, and compared the performance of the learning robot with/without proposed algorithms. Our results show that adopting proposed algorithms improves the robot's performance in terms of learning efficiency, complexity of the learned behaviors, predictability of the robot, and even the result of the subjective evaluation of the participants about the intelligence of the robot as well as the quality of the interaction.

Keywords: contingency evaluation, developmental robot, ostensive cue, human-robot interaction, joint attention

1. INTRODUCTION

Joint attention related behaviors (JARBs) include basic social skills, such as following the gaze of others, pointing, intention sharing, and social referencing. Humans gradually learn these social skills during their developmental process in infancy and childhood (Scaife and Bruner, 1975; Adamson, 1995; Corkum and Moore, 1995), and become able to establish interaction with others. Consequently, children become able to learn more social skills, such as language communication and mind reading (Moore and Dunham, 2014). The importance of JARBs in human infant development (Tomasello et al., 1995) has made it one of the most popular research topics in the fields of cognitive science and developmental psychology (Butterworth and Jarrett, 1991; Mundy et al., 2000; Tomasello, 2009).

Additionally, owing to the important role of such behaviors in achieving successful communication with humans, some robotic research has focused on the study of JARBs in the development of communicative robots (Imai et al., 2003; Breazeal, 2004; Kanda et al., 2004; Kaplan and Hafner, 2006).

On the other hand, in the field of developmental robotics, several studies based on synthetic approaches have tried to explore and/or reproduce the developmental process of the human infant, as well as to create autonomous developmental robots. See Asada et al. (2009) for a review of these efforts. Some of these research has been done on proposing learning mechanisms based on the intrinsic motivation of the robot that enables open-ended development (Oudeyer et al., 2007; Barto, 2013; Nehmzow et al., 2013), and some on dynamic Bayesian networks to evaluate the contingency of the observed events, which enables the robot to plan suitable action(s) to achieve its goal utilizing the evaluated contingency (Degris et al., 2006; Jonsson and Barto, 2007; Mugan and Kuipers, 2012).

Other studies (Nagai et al., 2003; Triesch et al., 2006) have tried to explain the developmental process of the JARBs of the human infant by using an infant robot. They have focused on the causality of the infant robot's observations, actions and consequent experiences during interaction with a human caregiver. They showed that learning of the causal sensorimotor mapping from gaze patterns of the caregiver to the motor commands of the robot lead the robot to acquire a primitive JARBs, such as gaze following. However, the robot had *a priori* knowledge of the set of sensory and motor variables to be associated in order to acquire such a sensorimotor mapping.

Sumioka et al. proposed an informational measure based on transfer entropy (Schreiber, 2000), by which the robot become able to automatically distinguish the set of sensory-motor variables for the sensorimotor mapping without such *a priori* knowledge (Sumioka et al., 2010). Additionally, their presented method could evaluate the contingency of a sequence of events, so that the robot became able to learn a sequence of sensorimotor mapping. The contingency of such sequence was defined as *contingency chain* (c-Chain). By using computer simulation, they showed that evaluating the c-Chains of the events led their infant robot model to learn JARBs consisted of sequences of actions, such as *social referencing* behavior. The social referencing was defined as looking back at the caregiver's face after producing the gaze-following behavior. Hereafter, we refer to robot's learned behavior as a *complex skill* if it consists of more than two sequences of actions (such as social referencing behavior), and otherwise refer to it as a *simple skill* (such as gaze-following behavior).

However, numerous time steps were required for the contingency evaluations of previous work (Sumioka et al., 2010), especially for complex skills, which resulted in the robot not being able to acquire complex skills in the real-world implementation (Sumioka et al., 2013). Mahzoon et al. (2016) proposed a new informational measure based on what they called *transfer information*, which enabled the local evaluation of the contingency among the variable values. They realized a fast contingency evaluation, even with a small number of sample data. They showed that their infant robot model could acquire simple and complex skills within short periods of interaction with the caregiver model, in a computer simulation environment.

Nevertheless, to implement the proposed method on a real-world robot, two basic issues are still remained: First, the synchronization problem of the robot's learning phase with the human caregiver's teaching phase in the real-world interaction was not considered. As a result, the efficiency of the learning process was decreased and therefore unexpectedly delayed. Although understanding and detecting the teaching phase of the human caregiver is not a simple issue, some research on "natural pedagogy" has reported the phenomena of teaching/learning timing of the human caregiver/infant (Csibra and Gergely, 2009) and addressed "ostensive cues" as the key signals of efficient teaching/learning in humans. In this paper, we propose a new algorithm for robot learning inspired by these phenomena, namely ostensive-cue sensitive learning (OsL), to overcome the synchronization problem. Second, there was overestimation of the contingencies related to actions/observations that occur simultaneously with the usage of a learned behavior. This is due to the confusion of the robot about the cause of the consequent event; the robot could not distinguish whether the reason for the event was the usage of the learned behavior or simply the previous atomic action/observation. To solve this problem, we propose another new algorithm, the exclusive evaluation of policies (XEP), following which the robot evaluates contingencies, so that the calculations related to the atomic variables are separated from those of the learned behaviors.

To evaluate the performance of each proposed algorithm in a real-world environment, we conducted human–robot interaction experiments under four conditions: (1) the previous method (Mahzoon et al., 2016), i.e., the robot uses neither of the proposed algorithms; (2) the robot uses only the OsL; (3) the robot uses only the XEP; and (4) the proposed method, i.e., the robot uses both the OsL and XEP. Each condition was consisted of 12 subject experiments, and each experiment was taken 800 time steps, i.e., approximately 40 min of interaction with the robot. The performances of the systems was compared in terms of the speed, coverage, and reliability of simple and complex skill acquisition.

In addition, as described in Moore and Dunham (2014) and Tomasello (2009), contingent and intelligent behavior of the infant "induces" the caregiver to change its behavior, and teach new concepts to the infant. This inherent tendency of the human caregiver leads to a potential for the open-ended learning and development of the infant, even an infant robot (Oudeyer et al., 2007). In our experiment, to evaluate if/how the human subjects feel regarding the infant robot's such intelligence, we conducted a subjective evaluation during the experiment. We asked the subjective opinion of the caregivers about the intelligence of the robot as well as the quality of the interaction. For this, we provided seven questions, each designed with a five-level Likert scale answer. To see the effect of the proposed algorithms on the subjective evaluation, we conducted a statistical analysis of the answers. The result of the analysis is discussed in section 4.5.

2. PROBLEM SETTING AND CONTINGENCY EVALUATION

2.1. Interaction Environment

A face-to-face interaction between a human caregiver and an (infant) robot is assumed as our experimental environment (**Figure 1**). There is a table between them and one or more objects are placed on the table. The human caregiver plays and interacts with the robot (based on their own strategy, if any) and can move the position of the objects on the table. The robot discretizes time. At each time step t, the robot observes the environment and stores the observed data in the sensory variables $\boldsymbol{S}^t = (S_1^t, S_2^t, \cdots, S_{N_S}^t)^T$, where N_S denotes the number of sensory variables. We also refer to these by "state variable" in this paper. After the observation, it sends action commands to its joints and saves them to the action variables $\boldsymbol{A}^t = (A_1^t, A_2^t, \cdots, A_{N_A}^t)^T$, where N_A denotes the number of action variables, which would be equal to the number of the joints of the robot. Next, the robot observes the result of the taken action, and saves the resultant observations to the resultant variables: $\boldsymbol{R}^t = (R_1^t, R_2^t, \cdots, R_{N_R}^t)^T$ for the values of the resultant observation before taking the action, and $\boldsymbol{R}^{t+1} = (R_1^{t+1}, R_2^{t+1}, \cdots, R_{N_R}^{t+1})^T$ for after taking the action, where N_R denotes the number of the resultant variables. In the remainder of this section, we summarize and introduce the basic idea of the contingency evaluation mechanism of our previous work (Mahzoon et al., 2016).

2.2. Finding and Reproducing Contingency

Assume that in time step t, the robot observes s_i^t and r_k^t, takes the action a_j^t, and as result, observes r_k^{t+1}; here, s_i^t, a_j^t, r_k^t, and r_k^{t+1} indicate the values of the variables S_i^t, A_j^t, R_k^t, and R_k^{t+1}, respectively. The quaternion $\boldsymbol{e} = (s_i^t, a_j^t, r_k^t, r_k^{t+1})$ represents such an experience of the robot, and is simply denoted as *experience* in this paper. An experience $\boldsymbol{e}$ contains information about "*when* (s_i^t), *what to do* (a_j^t), *for which transition* (r_k^t to r_k^{t+1})." During the interaction with the human, the robot evaluates the "contingency" of its experiences, which will be described later, and distinguishes the "contingent" ones. After finding the contingent experience(s), the robot tries to "reproduce" it by acquiring a suitable sensorimotor mapping that enables the robot to take suitable action a_j^t in the specific state s_i^t to reproduce the specific transition of r_k^t to r_k^{t+1}. Inspired by previous works on human infant behaviors concerning the process of finding and reproducing interaction contingencies (Watson, 1972), even with a contingently responsive robot (Movellan and Watson, 2002), in our work, the ability to reproduce the contingency of an interaction is considered to be one of the most essential social skills for an interactional robot, which makes it able to interact properly with the interacting human.

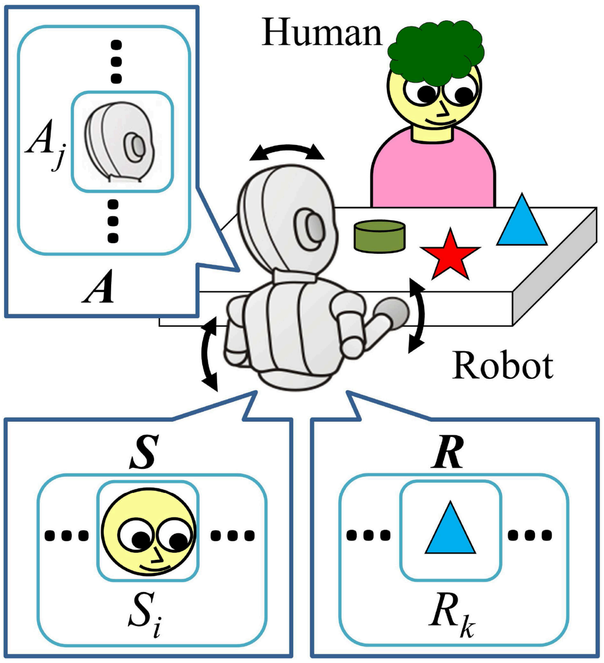


FIGURE 1 | Problem setting of the face-to-face interaction of the robot with a human caregiver. They sit across a table, and there are some objects on the table. The robot can produce actions such as moving head and hands as illustrated with the arrows in the figure. During the interaction with the human, the sensory data, taken actions, and resultant observations are stored in the sensory, action, and resultant variables, respectively ($\boldsymbol{S}$, $\boldsymbol{A}$, and $\boldsymbol{R}$).

To evaluate the contingency of the experiences, the robot updates and saves histograms of the values of the variables in each step of the interaction, and calculates the following probabilities. Assume there are two discrete-time stochastic processes X and Y, which can be approximated by stationary Markov processes. The transitions of the processes from time t to $t+1$ can be represented by the transition probabilities $p(x^{t+1}|x^t)$ and $p(y^{t+1}|y^t)$, where the notifications x^t, y^t and x^{t+1}, y^{t+1} indicate the values of the processes at times t and $t+1$, respectively. The contribution of a specific value of process Y, such as y^t, on the transition of the process X from a specific value such as x^t to a specific value x^{t+1} can be estimated using *transfer information* (Mahzoon et al., 2016):

$$I_{y \to x} = \log \frac{p(x^{t+1}|x^t, y^t)}{p(x^{t+1}|x^t)}. \tag{1}$$

For an experience $\boldsymbol{e}$, the transfer information can be adopted as follows to evaluate the contingency of the experience, i.e., the contribution of the action a_j^t in state s_i^t to the transition of r_k^t

to r_k^{t+1}, or in other words the joint contribution of the state and action in experience $\boldsymbol{e}$:

$$C_J(\boldsymbol{e}) = I_{(s_i,a_j)\to r_k} = \log \frac{p(r_k^{t+1}|s_i^t, a_j^t, r_k^t)}{p(r_k^{t+1}|r_k^t)}. \tag{2}$$

Additionally, the single contributions of the state and action in experience $\boldsymbol{e}$ can be calculated as follows:

$$C_S(\boldsymbol{e}) = I_{s_i\to r_k} = \log \frac{p(r_k^{t+1}|s_i^t, r_k^t)}{p(r_k^{t+1}|r_k^t)}, \tag{3}$$

$$C_A(\boldsymbol{e}) = I_{a_j\to r_k} = \log \frac{p(r_k^{t+1}|a_j^t, r_k^t)}{p(r_k^{t+1}|r_k^t)}. \tag{4}$$

The purpose of the robot is to evaluate the joint contribution in experiences to know if the action a_j^t in state s_i^t specifically leads to the consistent result r_k^{t+1}, and acquire a sensorimotor mapping of s_i^t to a_j^t. However, the value of Equation (2) can be also large when the value of the single contribution of either the state or action becomes large. Therefore, the joint contribution needs to be compared with the single contributions to distinguish the experiences in which the transition to r_k^{t+1} is due to both s_i^t and a_j^t, and not simply one of them. It can be estimated as follows:

$$\begin{aligned} {}^S\widetilde{C}_J(\boldsymbol{e}) &= C_J(\boldsymbol{e}) - C_S(\boldsymbol{e}) \\ &= \log \frac{p(r_k^{t+1}|s_i^t, a_j^t, r_k^t)}{p(r_k^{t+1}|s_i^t, r_k^t)}, \end{aligned} \tag{5}$$

$$\begin{aligned} {}^A\widetilde{C}_J(\boldsymbol{e}) &= C_J(\boldsymbol{e}) - C_A(\boldsymbol{e}) \\ &= \log \frac{p(r_k^{t+1}|s_i^t, a_j^t, r_k^t)}{p(r_k^{t+1}|a_j^t, r_k^t)}, \end{aligned} \tag{6}$$

where ${}^S\widetilde{C}_J(\boldsymbol{e})$ and ${}^A\widetilde{C}_J(\boldsymbol{e})$ compare the joint contribution with the single contribution of the state and action, respectively. Finally, the measure named *synergistic contribution of contingencies* (ScC) is proposed as follows to distinguish the "*contingent*" experiences, i.e., the experiences in which the combination of the state and the action is the cause of the transition, but not either of them is individually the cause:

$$\widetilde{C}_J(\boldsymbol{e}) = \min\{{}^S\widetilde{C}_J(\boldsymbol{e}), {}^A\widetilde{C}_J(\boldsymbol{e})\}. \tag{7}$$

When the value of $\widetilde{C}_J(\boldsymbol{e})$ of a specific experience $\boldsymbol{e}$ becomes larger than a specific threshold C_T for a predefined duration, such as θ time steps, the robot distinguishes it as a contingent experience (or simply, a contingency) and acquires the sensorimotor mapping (s_i^t, a_j^t). Then, it starts to "reproduce" the found contingency by "using" the acquired sensorimotor mapping. The sensorimotor mapping learned based on the experience $\boldsymbol{e}$ is denoted as the policy π. During interaction with the human, the robot may acquire several different policies. Note that θ is a parameter to determine how carefully the observed contingency is judged to be stable.

2.3. Evaluating the Contingency Chain

After the acquisition of a new m-th policy π_m, the robot adds a new Boolean variable S_{π_m} to the set of state variables, which indicates whether the policy π_m was used. It takes the value 1 if π was used, and 0 otherwise. To avoid confusion, we also denote the value of the S_{π_m} with $\bar{\pi}_m$ when it takes the value 0, and with π_m when it is 1. Then, the robot continues updating the histograms of the variables as well as calculating the contingency of the experiences, including the new state variable S_{π_m}. Using this method, the robot becomes able to evaluate the contingency of the c-Chains, and as a result, evaluate the contingency related to the new behavior of the caregiver who observed the contingency reproduction of the robot. In previous work (Mahzoon et al., 2016), an example of such a c-Chain was the consistent response of the caregiver to the social referencing behavior of the robot: the robot found that after using the gaze-following skill, if it looks at the caregiver's face, the caregiver will look at the face of the robot as an acknowledgement. Moreover, they showed that in a more complex simulation environment, the robot acquires a longer sequence of actions, up to five sequences.

3. PROPOSED METHOD

In this section, after discussing the two essential weak points of the previous work (Mahzoon et al., 2016) and our solution for each of them, we describe the mechanism of our proposed method.

3.1. Ostensive-Cue Sensitive Learning (OsL)

The first problem of previous work is the synchronization of the teaching phase of the human caregiver with the learning phase of the infant robot. Learning under the non-synchronized environment decreases the learning efficiency of the robot, and causes significant delays in the learning progress. Although distinguishing the teaching phase of the human by the robot seems to be a difficult issue owing to the probable variety of types of teaching in different human subjects, there are several reports in the fields of cognitive science and developmental psychology regarding how human infants treat the synchronization problem and increase the efficiency of learning from adults (see a review Csibra and Gergely, 2011).

Csibra and Gergely addressed the "natural pedagogy" as a human communication system for generic knowledge transmission between individuals (Csibra and Gergely, 2009). They proposed that human infants are "prepared to be at the receptive side of natural pedagogy" and sensitive to learn from the ostensive cues of human adults, such as mutual eye contact between the adults and the infant, or adults' infant-directed speech (motherese). From this statement, we hypothesize that the human adult may inherently or adaptively output the ostensive cues when it tries to teach something to the human infant, or even to the infant robot. Based on this hypothesis, we propose the OsL algorithm for the infant robot as follows: (1) The robot stops moving when it observes an ostensive cue from the human and continues the observation of the human until the signal disappears. This is because the ostensive cue acts as a signal (from our hypothesis) that informs the robot about the human's teaching phase, and notifies the robot to synchronize with it; (2) The robot counts the histogram of the consequent experiences right after the disappearance of the ostensive cue η times (i.e., the learning weight parameter of the OsL algorithm) instead of one time in order to emphasize such experiences. This is because (from our hypothesis) after the ostensive signals, the human would be in the teaching phase and the experiences right after the ostensive cues probably contain more informative concepts compared with other experiences. Using OsL, we expect the robot to increase the efficiency of learning and, as a result, the speed of skill acquisition.

3.2. Exclusive Evaluation of Policy (XEP)

The second problem of the previous work is the overestimation of the transition probabilities of the single contingencies, which leads to an underestimation of ${}^{S}\widetilde{C}_J$ and/or ${}^{A}\widetilde{C}_J$, i.e., Equations (5) and (6), when the robot uses an acquired policy. This leads to the underestimation of the ScC of some experiences, i.e., $\widetilde{C}_J$: Equation (7). The reasons for the overestimation and the underestimation are as follows. Assume that the robot acquired its new m-th policy π_m based on the contingent experience $\boldsymbol{e}_m = (s_i^t, a_j^t, r_k^t, r_k^{t+1})$. Before the robot starts to use π_m, i.e., using the sensorimotor mapping (s_i^t, a_j^t), the ${}^{S}\widetilde{C}_J$ and ${}^{A}\widetilde{C}_J$ of the experience $\boldsymbol{e}_m$ can be written by the transition probabilities calculated based on the histograms of the variables before acquiring and using π_m, i.e., p^{bef}, as follows:

$$ {}^{S}\widetilde{C}_J^{\text{bef}}(\boldsymbol{e}_m) = \log \frac{p^{\text{bef}}(r_k^{t+1}|s_i^t, a_j^t, r_k^t)}{p^{\text{bef}}(r_k^{t+1}|s_i^t, r_k^t)}, \tag{8} $$

$$ {}^{A}\widetilde{C}_J^{\text{bef}}(\boldsymbol{e}_m) = \log \frac{p^{\text{bef}}(r_k^{t+1}|s_i^t, a_j^t, r_k^t)}{p^{\text{bef}}(r_k^{t+1}|a_j^t, r_k^t)}. \tag{9} $$

However, when the robot starts to use π_m, the probability of taking action a_j^t in state s_i^t increases. This fact increases the value of the transition probabilities (1) $p(r^{t+1}|s_i^t, r_k^t)$ and (2)$p(r^{t+1}|a_j^t, r_k^t)$, i.e., the numerator of the single contingencies: Equations (3) and (4); and the denominator of ${}^{S}\widetilde{C}_J$ and ${}^{A}\widetilde{C}_J$: Equations (5) and (6). The reasons are (1) for $p(r^{t+1}|s_i^t, r_k^t)$: in state s_i^t, the probability of taking action a_j^t increases owing to the usage of π_m, which is a contingent skill and leads the transition to r_k^{t+1} with high probability; and (2) for $p(r^{t+1}|s_i^t, r_k^t)$: the probability of having been in state s_i^t when the action a_j^t is taken increases owning to the usage of π_m. Assume that the values of the transition probabilities $p(r^{t+1}|s_i^t, r_k^t)$ and $p(r^{t+1}|a_j^t, r_k^t)$ after the usage of π_m, i.e., denoted by p^{aft}, increase by factors of α and β, respectively, compared to p^{bef}:

$$ p^{\text{aft}}(r_k^{t+1}|s_i^t, r_k^t) = \alpha .\, p^{\text{bef}}(r_k^{t+1}|s_i^t, r_k^t)\ ;\ \alpha > 1 \tag{10} $$

$$ p^{\text{aft}}(r_k^{t+1}|a_j^t, r_k^t) = \beta .\, p^{\text{bef}}(r_k^{t+1}|a_j^t, r_k^t)\ ;\ \beta > 1 \tag{11} $$

Assuming that the value of the transition probability $p(r_k^{t+1}|s_i^t, a_j^t, r_k^t)$ does not change before and after the usage of π_m (because the usage of π_m as a sensorimotor mapping (s_i^t, a_j^t) is included in the condition part of the transition probability), the values of ${}^{S}\widetilde{C}_J$ and ${}^{S}\widetilde{C}_J$ for the experience $\boldsymbol{e}_m$ after the usage of π_m can be written as:

$$ \begin{aligned} {}^{S}\widetilde{C}_J^{\text{aft}}(\boldsymbol{e}_m) &= \log \frac{p^{\text{bef}}(r_k^{t+1}|s_i^t, a_j^t, r_k^t)}{\alpha .\, p^{\text{bef}}(r_k^{t+1}|s_i^t, r_k^t)} \\ &= {}^{S}\widetilde{C}_J^{\text{bef}}(\boldsymbol{e}_m) - \log\alpha \qquad ;\ \alpha > 1, \end{aligned} \tag{12} $$

$$ \begin{aligned} {}^{A}\widetilde{C}_J^{\text{aft}}(\boldsymbol{e}_m) &= \log \frac{p^{\text{bef}}(r_k^{t+1}|s_i^t, a_j^t, r_k^t)}{\beta .\, p^{\text{bef}}(r_k^{t+1}|a_j^t, r_k^t)} \\ &= {}^{A}\widetilde{C}_J^{\text{bef}}(\boldsymbol{e}_m) - \log\beta \qquad ;\ \beta > 1. \end{aligned} \tag{13} $$

Therefore, ScC of the experience $\boldsymbol{e}_m$ after the usage of the π_m will become:

$$ \begin{aligned} \widetilde{C}_J^{\text{aft}}(\boldsymbol{e}_m) &= \min\{{}^{S}\widetilde{C}_J^{\text{bef}}(\boldsymbol{e}_m) - \log\alpha,\ {}^{A}\widetilde{C}_J^{\text{bef}}(\boldsymbol{e}_m) - \log\beta\} \\ &< \widetilde{C}_J^{\text{bef}}(\boldsymbol{e}_m). \end{aligned} \tag{14} $$

To avoid such an underestimation, we propose to separate the contingency evaluations related to the acquired policies and atomic variables, namely the XEP algorithm. In this algorithm, the system adds an *extra* variable for each sensory and action variable to the system, denoted by $\widehat{S}_i^t$ and $\widehat{A}_j^t$. When an acquired policy π_m is used, the system sets the values of $\widehat{S}_i^t$ and $\widehat{A}_j^t$ to *don't care*. Therefore, the histogram of the values of these variables, denoted by $\hat{s}_i^t$ and $\hat{a}_j^t$, are counted only if an acquired policy has not been used. Using the histogram of these variables for the calculation of the transition probabilities of Equations (10) and (11), which are denoted by $\hat{p}$, causes them not to increase even after usage of the policy π_m:

$$ \begin{aligned} \hat{p}^{\text{aft}}(r_k^{t+1}|s_i^t, r_k^t) &= p^{\text{aft}}(r_k^{t+1}|\hat{s}_i^t, r_k^t) \\ &= p^{\text{bef}}(r_k^{t+1}|s_i^t, r_k^t), \end{aligned} \tag{15} $$

$$ \begin{aligned} \hat{p}^{\text{aft}}(r_k^{t+1}|a_j^t, r_k^t) &= p^{\text{aft}}(r_k^{t+1}|\hat{a}_j^t, r_k^t) \\ &= p^{\text{bef}}(r_k^{t+1}|a_j^t, r_k^t). \end{aligned} \tag{16} $$

Therefore, when the XEP algorithm is used, the value of ${}^{S}\widetilde{C}_J$ and ${}^{A}\widetilde{C}_J$ for the experience $\boldsymbol{e}_m$, which are denoted by ${}^{S}\widehat{C}_J$ and ${}^{A}\widehat{C}_J$, after the usage of π_m will be:

$$ \begin{aligned} {}^{S}\widehat{C}_J^{\text{aft}}(\boldsymbol{e}_m) &= \log \frac{p^{\text{bef}}(r_k^{t+1}|s_i^t, a_j^t, r_k^t)}{\hat{p}^{\text{aft}}(r_k^{t+1}|s_i^t, r_k^t)} \\ &= {}^{S}\widehat{C}_J^{\text{bef}}(\boldsymbol{e}_m), \end{aligned} \tag{17} $$

$$ \begin{aligned} {}^{A}\widehat{C}_J^{\text{aft}}(\boldsymbol{e}_m) &= \log \frac{p^{\text{bef}}(r_k^{t+1}|s_i^t, a_j^t, r_k^t)}{\hat{p}^{\text{aft}}(r_k^{t+1}|a_j^t, r_k^t)} \\ &= {}^{A}\widehat{C}_J^{\text{bef}}(\boldsymbol{e}_m). \end{aligned} \tag{18} $$

As the result, the ScC of the experience $\boldsymbol{e}_m$ when the XEP algorithm is used, which is denoted by $\widehat{C}_J$, after the usage of π_m

will be:

$$\begin{aligned}\widehat{C}_J^{\mathrm{aft}}(\boldsymbol{e}_m) &= \min\{{}^S\widehat{C}_J^{\mathrm{aft}}(\boldsymbol{e}_m),\ {}^A\widehat{C}_J^{\mathrm{aft}}(\boldsymbol{e}_m)\}\\ &= \widehat{C}_J^{\mathrm{bef}}(\boldsymbol{e}_m).\end{aligned} \tag{19}$$

With respect to Equation (19) and Inequation (14), it can be concluded that the XEP algorithm is able to solve the underestimation problem of the previous work (Mahzoon et al., 2016), and is expected to increase the accuracy of the contingency evaluation[1].

3.3. Mechanism

Figure 2 shows the schema of the proposed system. It consists of two main parts: the Contingency Detection Unit (CDU) and the Action Producing Unit (APU). The APU is responsible for determining the output action in each time step, while the CDU evaluates the contingency of the experiences. At each time step t, the robot observes the environment and stores the results of the current observation in $\boldsymbol{S}^t$ and $\boldsymbol{R}^t$ (bottom part of the figure). They are sent to the APU, and the APU decides about the outputting action for each joint of the robot $\boldsymbol{A}^t$, based on the input data $\boldsymbol{S}^t$ and $\boldsymbol{R}^t$. After taking the action, the robot again observes the environment, and stores the resultant observation in the resultant variable $\boldsymbol{R}^{t+1}$ (bottom part of the figure). Simultaneously, in each time step, the CDU gets the values of all of the variables, and evaluates the contingency of the experiences. If the CDU detects an experience as a contingent one, it adds a new Contingency Reproducer (CR in **Figure 2**) to the APU, which enables the APU to reproduce the found contingency. In the remainder of this section, each component of the CDU and APU are explained in detail.

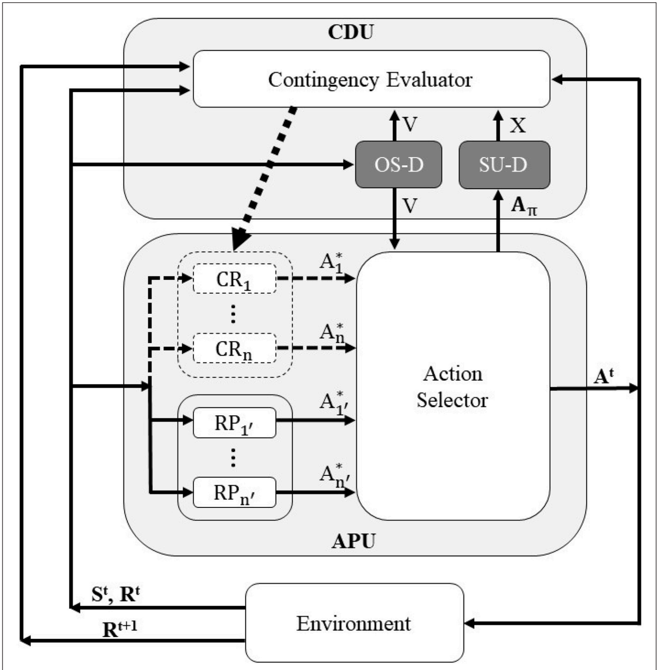


FIGURE 2 | System schema of the proposed mechanism, consisting of two main parts: Contingency Detection Unit (CDU) and Action Producing Unit (APU). Contingency Reproducer (CR), Reaction Producer (RP), and Action Selector compose the APU, while the Contingency Evaluator, Ostensive Signal Detector (OS-D) and Skill Usage Detector (SU-D) form the CDU. The new components of the proposed mechanism are shown with the darker color (OS-D and SU-D). V, X, and A_π indicates the controlling signals described in section 3.3.1. In each time step, the robot outputs the action A^t based on its current states S^t and R^t, and observes the resultant transition of the environment, i.e., R^{t+1}.

3.3.1. Contingency Detection Unit (CDU)

In each time step, the CDU (1) evaluates the contingency of the experiences, and (2) if a contingent experience is detected, it adds a new CR to the APU, which enables the robot to reproduce the found contingency. The CDU consists of three components: the Contingency Evaluator, Ostensive Signal Detector (OS-D), and the Skill Usage Detector (SU-D).

3.3.1.1. Contingency Evaluator

This unit calculates the contingencies of the experiences based on the histograms of the experiences, using the method described in section 2.2. If the experience $\boldsymbol{e} = (s_i^t, a_j^t, r_k^t, r_k^{t+1})$ is distinguished as a contingent one, it adds a new CR to the APU, which contains the values of the variables of the found contingent experience $\boldsymbol{e}$, i.e., s_i^t, a_j^t, r_k^t and r_k^{t+1}. After that, the Contingency Evaluator continues the evaluation of the contingencies, including the c-Chains (see section 2.3), as well as the process of adding further CRs to the system.

3.3.1.2. OS-D

The OS-D gets the current state of the robot (S_i^t and R_k^t). If it detects that these variables include an ostensive cue from the human, it sends the *stop signal V* to the Contingency Evaluator as well as the Action Selector. This signal causes the Contingency Evaluator to pause counting the histograms, and the Action Selector to make the robot to keep looking at the human and stop its movement. Additionally, it sends the learning weight parameter η (see section 3.1) to the Contingency Evaluator. When the ostensive cue disappears, the stop signal V is canceled simultaneously, which makes the Contingency Evaluator and Action Selector restart their functions. In this paper, mutual eye contact with the human caregiver is implemented as the only ostensive cue of the interaction.

3.3.1.3. SU-D

The SU-D gets the information regarding the usage of the policies in each time step from the Action Selector, and informs the Contingency Evaluator if any policy has been used at the current moment. To this end, the SU-D gets the values of the Boolean variable A_{π_m} from the the Action Selector, which indicates if the m-th policy is currently used, and sends the Boolean signal X to the Contingency Evaluator, which is calculated as follows:

$$X = \bigvee_{m=1}^{N_\pi} A_{\pi_m}\ , \tag{20}$$

[1]For the same reason, the system also uses the *extra* variables $\widehat{S}_i^t$ and $\widehat{A}_j^t$ when the robot has used the policy in the previous time step, i.e., when $S_{\pi_m} = 1$.

where N_π denotes the number of the policies that the robot has acquired until now. If the value of the signal X is true, the Contingency Evaluator sets the value of the extra variables $\widehat{S}_i^t$ and $\widehat{A}_j^t$ to *don't care*, as described in section 3.2.

3.3.2. Action Producing Unit (APU)

As shown in **Figure 2**, the APU is equipped with three components, the Reaction Producers (RP), Contingency Reproducers (CR), and Action Selector. At the beginning of the interaction, the APU contains no CRs and selects the actions of the robot at each time steps from the suggested actions of the RPs, denoted by $A^*_{1'}$ to $A^*_{n'}$ in **Figure 2** where n' indicates the number of RPs in the system. Continuing the interaction with the caregiver leads the CDU to find contingent experiences and add CRs to the APU, which include specific sensorimotor mappings, as described in section 3.3.1. Similar to the RPs, the CRs send their suggested actions to the Action Selector, denoted by A^*_1 to A^*_n in **Figure 2**, where n indicates the number of CRs acquired by the robot. Therefore, after adding CRs to the system, the Action Selector needs to choose the outputting action command to each joint of the robot from all of the candidates: $A_m \in \{A^*_1, A^*_2, \cdots, A^*_n, A^*_{1'}, A^*_{2'}, \cdots, A^*_{n'}\}$ where m indicates the m-th joint of the robot.

3.3.2.1. Contingency Reproducer (CR)

The CR gets the current state of the robot and outputs its suggested action to the Action Selector, based on its sensorimotor mapping. Additionally, it sends the reliability Z to the Action Selector, which indicates the certainty of the transition to the expected state if the Action Selector selects its suggested action as the output action of the robot. Assume the m-th CR was added to the system based on the contingent experience $\boldsymbol{e}_m = (s_i^t, a_j^t, r_k^t, r_k^{t+1})$. If the current state S_i^t and R_k^t are the same as s_i^t and r_k^t of the CR, it outputs the candidate action a_j^t to the Action Selector. Otherwise, it does not send any candidate. In this paper, the CR sends the ScC of the experience $\boldsymbol{e}_m$, i.e., $\widehat{C}_J(\boldsymbol{e}_m)$, as its reliability Z_m to the Action Selector.

3.3.2.2. Reaction Producer (RP)

The RP gets the current state of the robot and outputs a pre-programmed reaction, which is sent to the Action Selector as the suggested action of the RP. Also it sends a constant value α_m as its reliability Z_m to the Action Selector, where m indicates the m-th RP. For the sake of simplicity, in this paper we considered only one RP for the system, which outputs a random action for any input state.

3.3.2.3. Action Selector

The Action Selector chooses the output action for each joint of the robot at each time step. A soft-max action selection was utilized to choose the output from the candidates. Assume that for the j-th joint of the robot, the number of RPs and CRs which send the candidate action to the Action Selector, namely inputting components, are N_j^R and N_j^C, respectively. At each time step, the probability of selecting the suggested action of the inputting component i for the joint j is calculated based on their reliability as follows:

$$P_i^j = \frac{\exp(Z_i/\tau)}{\sum_{k \in N_j^R + N_j^C} \exp(Z_k/\tau)}, \tag{21}$$

where Z_i indicates the reliability of the inputting component i, and τ is a temperature constant. Note that if Z_i is less than the omission threshold C_O, the Action Selector does not consider the inputting component i in Equation (21) and P_i^j for that component is set to zero. This mechanism enables the robot to have a chance to omit any acquired skill, which might be acquired owing to the noise, lack of sufficient experiences, or other error factors. We set $C_O = C_T - \varepsilon$, where the C_T is the skill acquisition threshold (see section 2.2), and ε is a constant value. Additionally, when more than two CRs with the same suggested action and different c-Chain length exist in the inputting components, the Action Selector considers only the one with the longer c-Chain length as the inputting component, and ignores the others, i.e., sets their P_i^j values to zero.

When the suggested output of the m-th CR with the policy π_m is selected as the output, the Action Selector sets the value of the Boolean variable A_{π_m} to 1. It sends A_{π_m} to the SU-D in each time step to inform the SU-D about the usage of the skills. Also, when the Action Selector gets the stop signal V from the OS-D, it stops outputting new action commands to the joints of the robot until the stop signal disappears.

4. EXPERIMENT AND RESULT

In this section, the results of the real-world robot experiment with human subjects are reported. To evaluate the effect of the proposed methods, i.e., the XEP and OsL algorithms, the performances of four different learning mechanisms are compared, of which the CDU consists of (1) neither the SU-D nor the OS-D, (2) only the SU-D, (3) only the OS-D, and (4) both the SU-D and the OS-D. In the remainder of this paper, they are referred to as the previous method, XEP method, OsL method, and proposed method, respectively. This study was carried out in accordance with the recommendations of the ethics committee for research involving human subjects at the Graduate School of Engineering Science, Osaka University. The protocol was approved by the ethics committee for research involving human subjects at the Graduate School of Engineering Science, Osaka University. All subjects gave written informed consent in accordance with the Declaration of Helsinki.

4.1. Subjects, Apparatus, and Procedure

Figure 3 shows the environment of the experiment, which was designed based on the concepts explained in section 2.1 and **Figure 1**. The human subject was asked to sit opposite the humanoid infant robot and interact with it naturally, as when he/she interacts with a human infant. The subject was asked to play with the robot using a toy on the table and draw the attention of the robot to the toy by teaching the current position of the toy as well as the name, color, shape, or other features of it. It is explained to the subject that the robot may learn some social skills from the behavior of the subject, and start to use them. When

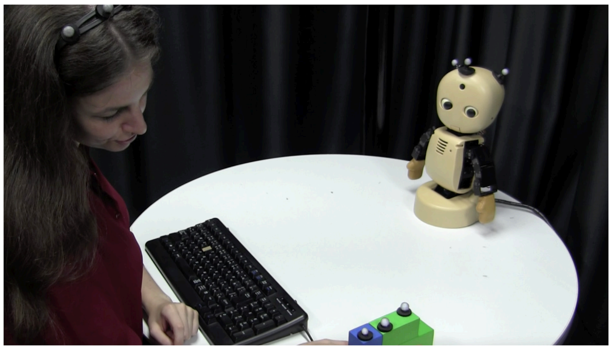


FIGURE 3 | The environment of the subject experiment. The subjects were asked to teach the current position of the toy to the robot. Also, they were asked to push a button of the keyboard to express that they are smiling and praising the robot at the moment. The consent for publication of this image was obtained from the participant of this image by using a written informed consent.

the robot uses a learned skill, the LEDs on the face of the robot turn on temporarily. The subject was asked to praise the robot by hitting a specific key on the keyboard when the robot finds the toy by using an acquired skill, i.e., when the LEDs turn on. Additionally, he/she was asked to change the position of the toy around every 20 s. The experiment was conducted for 800 time steps, i.e., around 40–50 min of interaction. After every 200 steps, i.e., around 10 min, the experiment was paused and the subject was asked to answer a simple questionnaire about the interaction, which may take <2 min (see section 4.5).

Twelve sessions were conducted for each four conditions described in section 4 using different human subjects, i.e., totally 48 adults: 30 males and 18 females. Before the main experiment, a test phase of 2 min was conducted to make everything clear for the subject. In this experiment, each time step was set to 2–2.5 s based on the complexity of the robot's internal calculations. Additionally, when the robot used a complex skill, the LEDs were set to temporally *flash* with frequency of $f = 2$Hz instead of just turning on; but the subject was not told about it.

4.2. Variables and Parameters

In this experiment, the number of objects was set to 1, and the position of the object on the table was quantized to 3 regions: left side, right side, and the middle of the table. Based on our experience, the other parameters were set as follows: for the CDU, $(C_T, \theta, \eta) = (0.7, 5, 2)$, and for the APU, $(\alpha_m, \tau, \varepsilon) = (0, 0.4, 0.1)$.

Table 1 shows the initial variables used in this experiment. For the perception $\boldsymbol{S}$, two variables were prepared: the gaze direction of the caregiver (S_1) and the observation of the object (S_2). S_1 takes the values f_1, f_2, and f_3 when the robot recognizes that the caregiver is looking at the left, right, and the middle of the table, respectively. It takes the value f_r when the robot detects that the caregiver is looking at it, and the value f_ϕ when the robot cannot detect the direction of the gaze of the caregiver. S_2 takes the value o when the robot detects the object, and o_ϕ when no object is detected. A motion capture system was utilized to detect the gaze direction of the caregiver as well as the position of the object in each time step.

TABLE 1 | Variables of the robot for the experiment.

Type	Variable name	Symbol	Elements
$\boldsymbol{S}$	Caregiver's gaze direction	C	$S_1 = \{f_1, f_2, f_3, f_r, f_\phi\}$
	Object	O_S	$S_2 = \{o, o_\phi\}$
$\boldsymbol{A}$	Gaze shifting	G	$A_1 = \{g_1, g_2, g_3, g_c\}$
	Hand Gesture	H	$A_2 = \{h_1, h_2, h_3, h_4\}$
$\boldsymbol{R}$	Frontal face of caregiver	F	$R_1 = \{\bar{r}_1, r_1\}$
	Profile of caregiver	P	$R_2 = \{\bar{r}_2, r_2\}$
	Object	O_r	$R_3 = \{\bar{r}_3, r_3\}$
	Praise from caregiver	W	$R_4 = \{\bar{r}_4, r_4\}$

For the actions of the robot $\boldsymbol{A}$, two variables were prepared: gaze shift (A_1) and the hand gesture of the robot (A_2). A_1 takes the values g_1, g_2, and g_3 when the robot shifts its gaze and looks at the left, right, and the middle of the table, respectively. It takes the value g_c when the robot looks at the caregiver's face. A_2 takes the values h_1, h_2, h_3, and h_4, which indicate the different types of hand gestures known by the robot. In this experiment, each values of the h_j were implemented as a different degree of the pitch of the robot's arm.

For the resultant perception $\boldsymbol{R}$, four Boolean variables were considered: the frontal face of the caregiver (R_1), the profile (face) of the caregiver (R_2), the observation of the object (R_3), and the praise from the caregiver(R_4). They take the value 1 if the frontal face, the face in profile, the object and the smile of the caregiver are observed by the robot. Otherwise, they take the value 0. To avoid confusion, the values of R_1, R_2, R_3, and R_4 are also denoted with r_1, r_2, r_3, and r_4 when they take 1, and with $\bar{r}_1$, $\bar{r}_2$, $\bar{r}_3$, and $\bar{r}_4$ when they are 0, respectively. In the experiment, to detect the values of R_1, R_2, and R_3, the motion capture system was utilized, while the praise from the caregiver, i.e., R_4, was expressed by the caregiver hitting a specific key on the keyboard. Also, to avoid confusion of the variables and to facilitate further discussions, each variable is mentioned with the symbol indicated in **Table 1** in the remainder of the paper.

4.3. Developmental Process of Social Skill Acquisition

Before the statistical comparison of performance of the different methods, we first show the developmental process of social skill acquisition by the robot using some examples from the experimental results of three subjects. **Tables 2–4** show the acquired skills by the robot during the experiment with these subjects, namely sbj-A, sbj-B, and sbj-C, respectively. While the robot utilized the previous method in the case of sbj-A, it used the proposed method for the case of sbj-B and sbj-C. Additionally, **Figure 4** shows the time course of the evaluated amount of contingencies related to each acquired skills indicated in **Tables 2–4** .

In these tables, the "ID" column indicates the ID of the contingency reproducer (CR),"Step" indicates the time-step

TABLE 2 | Acquired social skills by the robot for the sbj-A.

ID	Step	Level	Label	r^t	s^t	a^t	r^{t+1}	Interpreted function
π_1	101	1	**Gaze-Following-2**	$\bar{r}_3$	f_2	g_2	r_3	Gaze Following (middle)
π_2	340	1	**Gaze-Following-1**	$\bar{r}_3$	f_1	g_1	r_3	Gaze Following (right)
π_3	370	1	**Gaze-Following-0**	$\bar{r}_3$	f_0	g_0	r_3	Gaze Following (left)
π_4	519	2	**Looking-Back-2**	$\bar{r}_4$	π_1	g_c	r_4	Looking Back (after Gaze-Following-2)

TABLE 3 | Acquired social skills by the robot for sbj-B.

ID	Step	Level	Label	r^t	s^t	a^t	r^{t+1}	Interpreted Function
π_1	191	1	**Gaze-Following-2**	$\bar{r}_3$	f_2	g_2	r_3	Gaze Following (middle)
π_2	295	1	**Gaze-Following-1**	$\bar{r}_3$	f_1	g_1	r_3	Gaze Following (right)
π_3	418	1	**Gaze-Following-0**	$\bar{r}_3$	f_0	g_0	r_3	Gaze Following (left)
π_4	485	2	**Looking-Back-0**	$\bar{r}_4$	π_3	g_c	r_4	Looking Back (after Gaze-Following-0)
π_5	611	2	**Looking-Back-1**	$\bar{r}_4$	π_2	g_c	r_4	Looking Back (after Gaze-Following-1)
π_6	655	2	**Looking-Profile-0**	$\bar{r}_2$	π_3	g_c	r_2	Finding Profile (after Gaze-Following-0)

TABLE 4 | Acquired social skills by the robot for the sbj-C.

ID	Step	Level	Label	r^t	s^t	a^t	r^{t+1}	Interpreted Function
π_1	100	1	**Gaze-Following-2**	$\bar{r}_3$	f_2	g_2	r_3	Gaze Following (middle)
π_2	129	1	**Gaze-Following-0**	$\bar{r}_3$	f_0	g_0	r_3	Gaze Following (left)
π_3	134	1	**Frontal-Face**	$\bar{r}_1$	o_ϕ	g_c	r_1	Finding Frontal Face
π_4	220	1	**Gaze-Following-1**	$\bar{r}_3$	f_1	g_1	r_3	Gaze Following (right)
π_5	372	2	**Looking-Back-1**	$\bar{r}_4$	π_4	g_c	r_4	Looking Back (after Gaze-Following-1)
π_6	512	1	**Hand-Motion**	$\bar{r}_3$	f_1	h_2	r_3	Finding Object by Hand
π_7	610	2	**Looking-Back-2**	$\bar{r}_4$	π_1	g_c	r_4	Looking Back (after Gaze-Following-2)
π_8	622	2	**Looking-Back-0**	$\bar{r}_4$	π_2	g_c	r_4	Looking Back (after Gaze-Following-0)
π_9	720	3	**Check-Again-1**	$\bar{r}_3$	π_5	g_1	r_3	Check Again the Object

at which that the CR was acquired, "Level" indicates the length of the c-Chain of the acquired CR, "Label" shows the symbol of the CR which may be used to refer to it by the subsequent CRs (and also it is used in **Figure 4** to indicate each CR), the column of "r^t, s^t, a^t, and r^{t+1}" indicate the experience $\boldsymbol{e}$ on which the CR was created, and finally, the interpretation of the CR is given based on the behavior of the robot when it uses the CR in the column of "Interpreted Function."

In **Figure 4**, the graphs of the simple and complex skills are separated: the top part (**Figures 4A–C**) for the simple skills and the bottom part (**Figures 4D–F**) for the complex ones. Each column of the figure indicates the result of each subject: from the left to right for sbj-A, sbj-B, and sbj-C, respectively. In each graph, the threshold of the contingency acquisition C_T is shown with the horizontal dotted gray line, and the hatched area indicates the values less than the threshold; while the vertical dashed lines indicate the time-step that each CR was acquired (the color is the

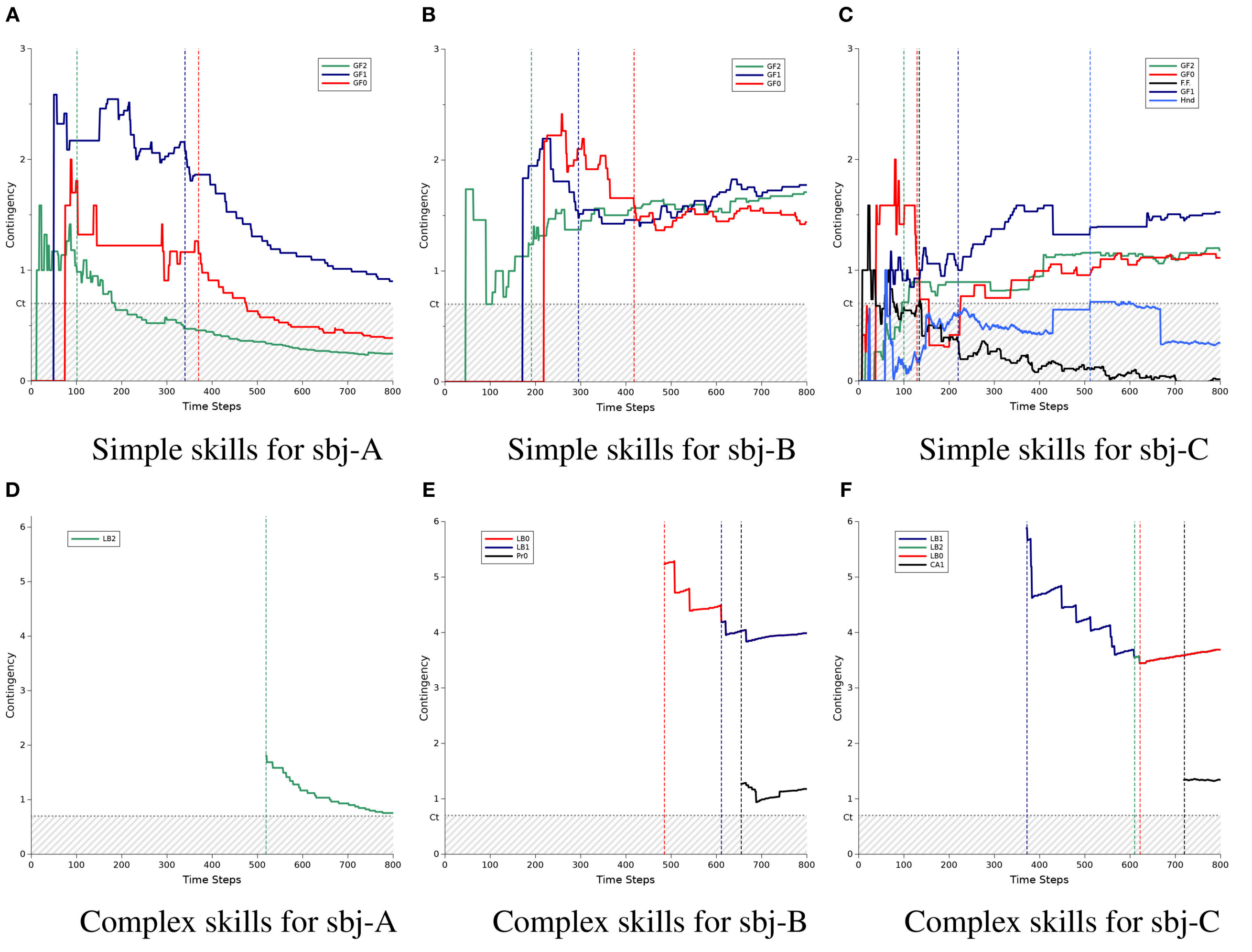


FIGURE 4 | Developmental process of the acquisition of social skills by the robot: a comparison among three participants (sbj-A, sbj-B, and sbj-C). For each subject, the process was shown for simple skills (top three graphs) and complex skills (bottom three graphs). The horizontal axes indicate the time step of the experiment (ends at 800, which is equal to around 40 min.), while the vertical axes labeled contingency indicate the amount of the calculated contingency using equation (7). Each sub-figure include the time courses of contingency for some sample set of experience $\boldsymbol{e}$, which are mentioned with a name such as Gaze-Following or Looking-Back in the legend of the figures. C_t indicated the threshold for the contingency of $\boldsymbol{e}$ to be acquired as a skill. The vertical dotted lines indicate the timing of the acquisition of a $\boldsymbol{e}$ as a skill, where its color represents which experience is acquired. Note that 2 lines (red and blue) in **(E)** and 3 lines (blue, green, and red) in **(F)** are overlapped, but represent contingency for different experiences. **(A)** Simple skills for sbj-A. **(B)** Simple skills for sbj-B. **(C)** Simple skills for sbj-C. **(D)** Complex skills for sbj-A. **(E)** Complex skills for sbj-B. **(F)** Complex skills for sbj-C.

same as that of the corresponding CR indicated in the legend of the graphs). Note that the order of the CRs at the legend of the graphs are the same as the order in which they were acquired. Also, the colors of the lines for Gaze-Following and Looking-Back are set based on their corresponding directions: red, blue, and green for the left, right, and the middle of the table, respectively.

According to the first row of **Table 2**, in the case of the sbj-A, where the robot was using the previous method, the robot acquired its first CR π_1 at $t = 101$, which for the inputs $(\bar{r}_3, f_2)$, outputs the action g_2 to observe r_3. In other words, this CR indicates that when the robot recognizes that the human subject is looking at the middle of the table (f_2), if the robot shifts its gaze to the same position, i.e., the middle of the table (g_2), then the robot can find the object (transition of $\bar{r}_3$ to r_3). Using this CR, the robot can produce the gaze following behavior (to the middle of the table). It is noted by the symbol **Gaze-Following-2** (where the number indicates the position of the table) and the time course of the calculated contingency of the experience related to **Gaze-Following-2**, i.e., $\boldsymbol{e}_{\mathrm{GF2}} = (f_2, g_2, \bar{r}_3, r_3)$, is shown in **Figure 4A** with the green line. From the beginning of the interaction, the contingency of **Gaze-Following-2** goes higher than the threshold C_T (the vertical dashed line), and after a while [namely, after experiencing the $\boldsymbol{e}_{\mathrm{GF2}}$ more than θ (=5) times], it is acquired as the first CR of the robot. The vertical green dashed line around $t = 100$ in **Figure 4A** shows the timing of the acquisition of this CR, which corresponds to the value of "Step" in π_1, **Table 2**. As shown in the figure, the value of the contingency of **Gaze-Following-2** was 0.98 at the acquisition time, while it decreases to 0.25 at the end of the experiment.

Following the time courses of the other contingencies in **Figure 4A** we can see that the robot acquired gaze-following skill to the right and left side of the table at $t = 340$ and $t = 370$, respectively (blue and red lines, corresponding with π_2 and π_3 of **Table 2**, respectively). After the acquisition of the skills, the robot starts to use them as described in section 3.3.2.3. At $t = 519$, the robot found a contingent relationship between using **Gaze-Following-2** and being praised by the human, and acquired new CR with a level of 2 (the green line in **Figure 4D** and π_4 in **Table 2**). This CR tells the robot that after using the gaze following to the middle of the table ($s^t = \pi_1$), if it shifts gaze to the human ($a^t = g_c$), then the robot would be praised by the human (transition of $r^t = \bar{r}_4$ to $r^{t+1} = r_4$). In this paper, we refer to this behavior as looking back behavior (**Looking-Back**). Acquisition of the **Looking-Back-2** would be due to the specific praising behavior of the human during the experiment (see section 4.1). This CR shows that the robot develops the acquired skills (such as **Gaze-Following-2**) to more complex ones (such as **Looking-Back-2**), which enables the robot to have longer interaction sequence with the human subject.

However, in the case of the sbj-A, the implemented method was the previous method. As described in section 3.2, the previous method has no mechanism to prevent the underestimation of contingencies after the acquisition of the CRs. Therefore, in **Figures 4A,D**, the contingency of the acquired CRs decreased after the acquisition of each CRs. As result, the contingency of the **Gaze-Following-2** and **Gaze-Following-0** (green and red lines) become less than the omission threshold C_O (=0.6), i.e., 0.1 lower than the threshold C_T in the graphs, and the Action Selector would stop using them. Additionally, a smaller value of the contingencies reduces the value of Z, which leads the Action Selector to use the CRs with less probability (see Equation 21). Therefore, in the previous method, although the robot could acquire simple and complex skills, it may not be able to use them properly.

Table 3, **Figures 4B,E** show the result of the experiment of sbj-B, in which the proposed method was implemented on the robot. Compared with the case of the sbj-A (which the previous method was implemented), the contingency of the **Gaze-Following**s do not decrease to less than (or close to) the omission threshold and, as a result, the robot could acquire more complex skills (two **Looking-Back**s and one **Looking-Profile**). Considering the probable irregular behavior of the human against the robot or the noise of the environment in the real-world interaction, preventing the underestimation of the contingencies seems to be very important, as shown in this example. Note that if the subjects had praised the robot when the robot found the object by using the Gaze-Following skill with high probability, the value of the contingency of **Looking-Back** is theoretically 4 with respect to Equation (7); assuming that the numerator of Equations (5) and (6) are approximately 1 due to the accurate praising behavior of the caregiver, while the denominator of Equation (5) is 0.25 because if the robot chooses the gaze action g_c from the four possible ones g_1,g_2,g_3, and g_c it would be praised, and the denominator of Equation (6) is at most 0.25 because it is equal to the probability that the robot had found the object before the robot takes the action g_c. During the experiment, although both the sbj-A and sbj-B seemed to praised the robot with same manner, the contingency of the **Looking-Back-2** (green line in **Figure 4D**) for the sbj-A became 0.76 at the end of the experiment, while in the case of the sbj-B, it became 3.99 for both **Looking-Back-0** and **Looking-Back-1** (red and blue lines in **Figure 4E**), which is very close to the value of the theoretical calculation. Note that the overlap of the **Looking-Back**s is due to the small number of the experiences related to the **Looking-Back**s, which makes the transition probabilities of their contingency evaluation very close to each other.

Following the time courses of **Figure 4E**, finally a new complex skill **Looking-Profile-0** is acquired. This CR (see π_6 of **Table 3**) causes the robot to look at the human (g_c) after following its gaze (π_3) to find human's face in profile (transition of $\bar{r}_2$ to r_2). This skill was specific to the sbj-B; it seems that he tended to show his face in profile to the robot when the robot succeeded to find the object by using the **Gaze-Following** skills, probably because he was concentrating to push the correct button of the keyboard to praise the robot while the keyboard was on the right side of the table in the case of the sbj-B. The acquisition of this kind of subject-specific skills shows that the proposed mechanism has the potential of evaluating various kind of human behaviors based on the different interaction manner of the subjects.

Figures 4C,F show the result of another subject, i.e., sbj-C, which the robot was implemented with the proposed method. The result shows more complex and interesting process of the contingency evaluation, acquisition, and omission by the robot. The details of the acquired skills are listed in **Table 4**. After acquiring the gaze-following skill to the middle and the left side of the table (**Gaze-Following-2** and **Gaze-Following-0**, the green and red lines in **Figure 4C**), the robot acquired a skill named **Frontal-Face** (the black line), which makes the robot to look at the human (g_c) to find his/her frontal face (r_1), when no object was detected (o_ϕ) at $t = 134$ (see π_3 in **Table 4**). However, finding the frontal face of the human is due to the single effect of the action g_c, but not the joint effect of the state o_ϕ and action g_c (see section 2.2 for the details of the single and joint effects). Therefore, as shown in the figure, the contingency of the **Frontal-Face** was reduced to less than the omission threshold and as a result, the **Frontal-Face** would not be selected by the Action Selector anymore. The acquisition and omission of this CR shows an example of how the proposed mechanism may acquire a non-contingent skill, use it, update the consequent of the usage of the skill, and finally recognize it as a non-contingent one and stop using it.

After the **Frontal-Face**, the robot acquired **Gaze-Following-1**, developed it to **Looking-Back-1**, and acquired another non-contingent skill named **Hand-Motion**, which indicates that the robot can find the object by hand gesture h_2. Since there seemed to be no relation between finding the object and the hand gestures of the robot, therefore the contingency of the **Hand-Motion** was reduced to less than the omission threshold after a while. Then, the robot acquired **Looking-Back-2** and **Looking-Back-0**, and finally acquired another complex skill with the level of 3, named "Check Again": **Check-Again-1**. This CR informs the robot after using **Looking-Back-1** (π_5), if it looks at the right side of the table (g_1), it can find the object again (r_3). In other words, when the robot detects that the human is looking at the right side of the table, it follows the gaze of the human and looks at the right side using **Gaze-Following-1** to find the object (π_4 in **Table 4**), then looks back at the human using **Looking-Back-1** to be praised (π_5 in the table), and then, looks at the right side again using **Check-Again-1** to see the object, again (π_9 in the table).

To summarize this section, we compared a result of one of the best cases of the previous method (sbj-A) with two cases from our proposed method: the case of sbj-B, in which the robot had a moderate performance and the case of sbj-C, in which the robot had a higher performance. In the cases of sbj-B and sbj-C, the robot was able to prevent the underestimation of the contingencies which occurred after the acquisition of the CRs in the previous method. This underestimation can be seen in the case of sbj-A. As a result, the robot could acquire more complex skills in these cases. This was due to the contribution of the XEP algorithm. Moreover, the averages of the time steps spent for the acquisition of simple and complex skills were smaller in these cases. This was due to the contribution of the OsL algorithm. The faster skill acquisition also resulted in the acquisition of more complex skills, concerning the limitation of the time in the real-world experiment.

4.4. Quantitative Analysis of Performance

In this section, the effect of the proposed algorithms on the performance of the system was explored. As the measure of the performance analysis, (1) the coverage of Gaze-Following, (2) the coverage of Looking-Back, (3) the time required to learn Gaze-Following, (4) the time required to learn Looking-Back, (5) the number of the acquired non-contingent skills, and (6) the number of the expected transition, was elected and the mean of each performance measure was compared among the experiment conditions. For each performance measure, a 2×2 ANOVA was conducted with two between subject factors OsL (0 or 1) and XEP (0 or 1), where 1 indicates that the algorithm was adopted and 0 indicates it was not. Also, a *post-hoc* power analysis was conducted to determine the observed power $(1-\beta)$ of the test, computed using $\alpha = 0.05$. In the following three sections, the definition of each performance measure, the result of the statistical tests, and the discussion about the result was proposed, respectively.

4.4.1. Performance Measure

For (1) the coverage of Gaze-Following and (2) the coverage of Looking-Back, the coverage of the acquired Gaze-Following and Looking-Back were calculated in terms of percentage, respectively, where 100% means that the robot learned the skill related to all positions: left, right, and middle of the table. With respect to the instructions of the experiment, the subjects would try to draw the attention of the robot to the object; therefore, the contingency of the Gaze-Following is expected to exist in the interaction, and had to be learned by the robot. Moreover, praising process of the caregiver would lead to the existence of the contingency of Looking-Back in the interaction and had to be learned by the robot, as well. Therefore, the coverage of Gaze-Following and Looking-Back seems be fair and adequate for comparing the learning performance of the systems; for the simple and the complex skills, respectively.

For (3) the time required to learn Gaze-Following and (4) the time required to learn Looking-Back, the average time steps required for learning Gaze-Following and Looking-Back for all three positions, i.e., left, right, and middle of the table; was considered, respectively. If a skill was not acquired, the value was set to 800, i.e., the total time of the experiment. These measures show the learning speed of the system, specifically the learning of the simple and complex skills, respectively.

On the other hand, the OsL uses weighted learning, which may increase the acquisition of the non-contingent skills; and the XEP may compensate it by increasing the accuracy of the contingency evaluation. For that, (5) the number of the acquired non-contingent skills, was considered to be compared among the conditions. These skills were defined as the ones apart from Gaze-Following, Looking-Back, Looking-Profile, and Check-Again. This measure is expected to reflect the non-efficiency of the learning mechanism of the robot.

Finally, the predictability of the learned skills was compared to evaluate the usability of the acquired skills of the robot. It was denoted as (6) the number of the expected transition; and calculated by the average number of the successful expected transitions of the environment conducted by utilizing the learned behaviors. For instance, if the Gaze-Following was used and as a result the robot could find the object, it was counted as a successful expected transition.

4.4.2. Result of Comparison and Test

The result of the performance comparison and ANOVA was summarized in **Figure 5**. In each graph of the figure, the average, and the standard deviation of the data gathered from the subject experiment was plotted. Additionally, the effect of each algorithm on the performance measure was denoted with the asterisk on the top left side of each figure, indicating the obtained p-value for the main effect of each algorithm by ANOVA[2]. The result of the mentioned two-way ANOVA for each of the performance measure is as follows.

For the coverage of Gaze-Following (**Figure 5A**), the ANOVA revealed a main effect of OsL, $F_{(1,44)} = 8.57$, $p = 0.005$, $\eta_p^2 = 0.163$, with $1-\beta = 0.846$, indicating that with using the OsL algorithm the coverage of Gaze-Following was higher ($M = 92.1\%$, $SD = 9.6$) than the case that the OsL was not used ($M = 74.8\%$, $SD = 35.8$). The significance was not confirmed neither for the main effect of XEP $F_{(1,44)} = 1.24$, $p = 0.27$, nor for the interaction between the OsL and XEP, $F_{(1,44)} = 0.14$, $p = 0.71$. Note that according to **Figure 5A**, the coverage of Gaze-Following was 69% ($SD = 33$) using the previous method, which increased to 81% ($SD = 39$) by applying the XEP, 94% ($SD = 13$) with the OsL, and 100% ($SD = 0$) using both of them as in the proposed method. The result of ANOVA for the coverage of Looking-Back (**Figure 5B**) showed a main effect of OsL, $F_{(1,44)} = 25.4$, $p < 0.001$, $\eta_p^2 = 0.366$, with $1-\beta =$

[2]The p-values are denoted by $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$, and ns, not significant, in the figures of this paper.

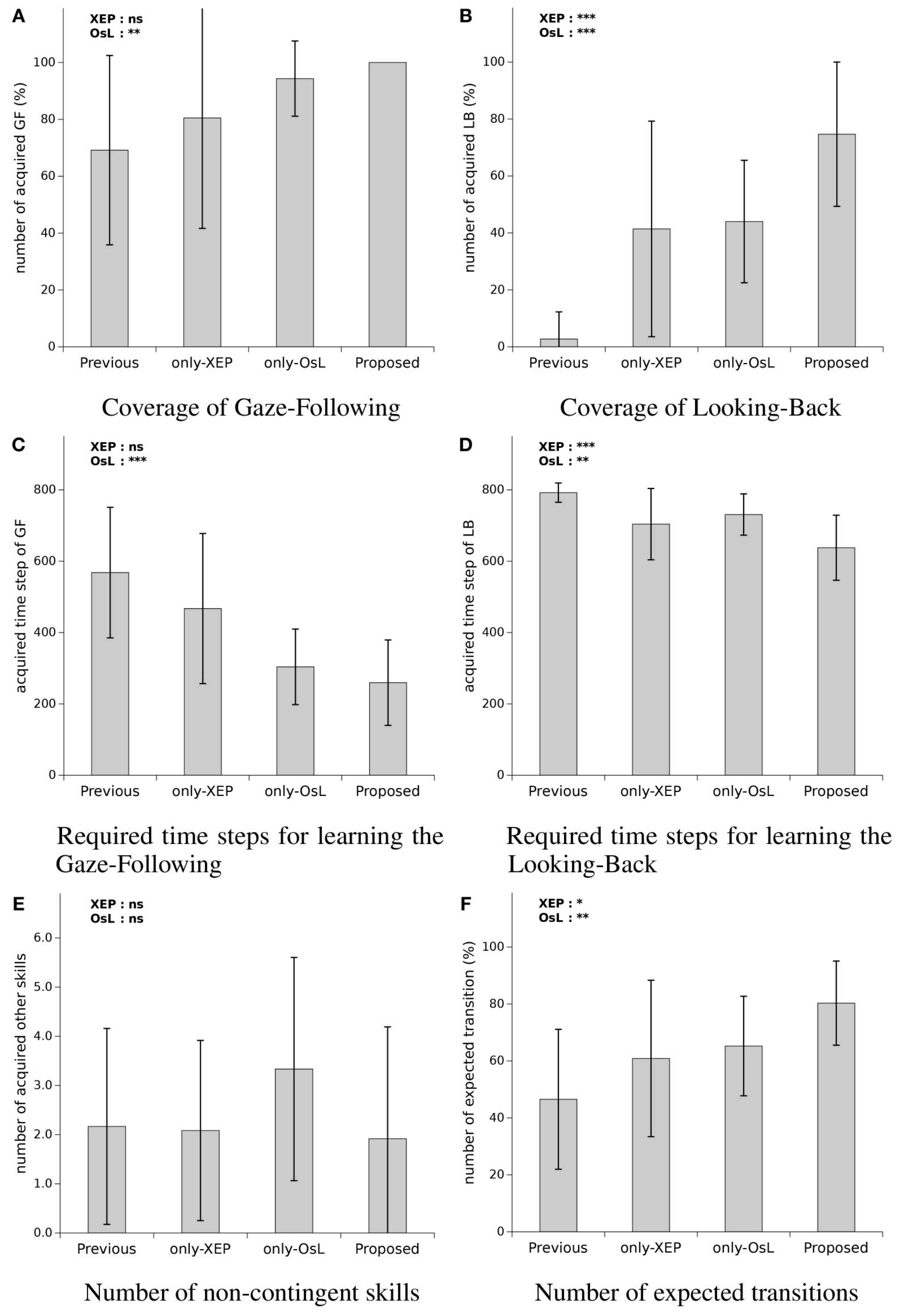


FIGURE 5 | Performance comparison of the four systems: **(A)** the number of learned skills labeled Gaze-Following, **(B)** the number of learned skills labeled Looking-Back (looking back), **(C)** spent time steps to acquire Gaze-Following, **(D)** spent time steps to acquire Looking-Back, **(E)** the number of the skills which is suppose to be not contingent but acquired, and **(F)** the number of transitions where the robot succeeded in observing a result as expected by using the acquired skills. At the top left side of each graph, significant levels of main effects in two-way ANOVA with OsL (Ostensive-cue sensitive Learning) and XEP (Exclusice Evaluation of Policy) as between-subject factors are mentioned. The *p*-values are denoted by $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$, and ns, not significant. Note that any interactions were not confirmed with the ANOVA.

0.999, indicating that the mean coverage of Looking-Back was greater when the OsL algorithm was adopted ($M = 59.3\%$, $SD = 27.8$) than the cases that the OsL was not used ($M = 22.1\%$, $SD = 33.4$). Also, the main effect of XEP yielded an F ratio of $F_{(1,44)} = 21.7, p < 0.001, \eta_p^2 = 0.333$, where $1 - \beta = 0.998$, indicating that the mean coverage of Looking-Back was higher by using the XEP algorithm ($M = 58.0\%$, $SD = 35.8$) than the cases that the XEP was not adopted($M = 23.4\%$, $SD = 26.6$). These main effects were not qualified by an interaction between OsL and XEP, $F_{(1,44)} = 0.29, p = 0.59$. Note that as mentioned in **Figure 5B**, the low performance of the previous method was improved from 3% ($SD = 10$) to 75% ($SD = 25$) by using the proposed method.

For the time required to learn Gaze-Following (**Figure 5C**), the main effect of OsL was confirmed with the ANOVA, $F_{(1,44)} = 25.9, p < 0.001, \eta_p^2 = 0.370$, with $1 - \beta = 0.999$, indicating that the mean time required for the acquisition of Gaze-Following was faster when the OsL algorithm was adopted ($M = 282$, $SD = 113$) compared to the cases that the OsL was not used ($M = 518$, $SD = 200$). However, the significance was shown neither for the main effect of XEP, $F_{(1,44)} = 2.45, p = 0.125$, nor for the interaction between the OsL and XEP, $F_{(1,44)} = 0.371$, $p = 0.55$. Note that as mentioned in **Figure 5C**, the time required to learn Gaze-Following became less than the half in the proposed method compared with the previous method, i.e., decreased from 568 steps ($SD = 183$) to 260 steps ($SD = 120$).

In the case of the time required to learn Looking-Back (**Figure 5D**), the result of ANOVA revealed a main effect of OsL, $F_{(1,44)} = 8.72, p = 0.005, \eta_p^2 = 0.165$, with $1 - \beta = 0.854$, indicating that the mean time for the learning of the Looking-Back was faster by using OsL ($M = 684$, $SD = 89$) than not using the OsL ($M = 748$, $SD = 85$). Also, the main effect of XEP yielded an F ratio of $F_{(1,44)} = 17.6, p < 0.001, \eta_p^2 = 0.286$, where $1 - \beta = 0.989$, suggesting that the mean time required to learn Looking-Back was faster when the XEP algorithm was adopted ($M = 670$, $SD = 100$) compered with the cases which the XEP was not used ($M = 760$, $SD = 54$). The significance was not confirmed for the interaction between the OsL and XEP, $F_{(1,44)} = 0.013, p = 0.91$. Note that the average time spent for the acquisition of the first Gaze-Following and Looking-Back skills by the robot using the proposed method was 8 min and 25 min with the standard deviation 5 and 7 min, respectively.

The result of the ANOVA for the number of the acquired non-contingent skills (**Figure 5E**) showed neither the main effect of OsL, $F_{(1,44)} = 0.68, p = 0.41$, nor the main effect of XEP, $F_{(1,44)} = 1.53, p = 0.22$, nor the interaction between the OsL and XEP, $F_{(1,44)} = 1.21, p = 0.28$. As mentioned in the figure, when only the OsL algorithm was utilized, it increased from 2.2 ($SD = 2.0$) to 3.3 ($SD = 2.3$), while adopting the XEP decreased it to 1.9 ($SD = 2.3$) with the proposed method. However, no significant effects of either of the algorithms were found in the result of the ANOVA for this measure. Finally, for the number of the expected transition (**Figure 5F**), the ANOVA revealed a main effect of OsL, $F_{(1,44)} = 9.28, p = 0.004, \eta_p^2 = 0.174$ with $1 - \beta = 0.875$, indicating that the mean number of the expected transition was grater when the OsL algorithm was adopted ($M = 72.8\%$, $SD = 17.6$) than the cases that the OsL was not used ($M = 53.7\%$, $SD = 26.5$). Also the main effect of XEP was supported by the ANOVA, $F_{(1,44)} = 5.51, p < 0.023, \eta_p^2 = 0.111$, where $1 - \beta = 0.669$, which suggests that the mean number of the expected transition was grater by using the XEP ($M = 70.6\%$, $SD = 23.7$) compared with the cases that the XEP was not implemented ($M = 55.9\%$, $SD = 22.9$). It is not confirmed for the interaction between the OsL and XEP, $F_{(1,44)} = 0.003, p = 0.96$. Note that according to the figure, the proposed method increased the number of the expected transition from 47% ($SD = 25$) to 80% ($SD = 15$).

4.4.3. Discussion

The OsL algorithm improved the coverage of Gaze-Following while both of the XEP and OsL algorithms improved the coverage of Looking-Back. Therefore, the XEP seems to be effective on learning complex skills, such as Looking-Back, while the OsL is useful to learn both complex and simple skills, such as Gaze-Following. The reason for these are considered to be the increased accuracy of the contingency evaluation (for XEP), and synchronizing the teaching/learning phases of the caregiver/robot (for OsL). Thus, adopting both of them will lead to the highest performance in terms of the coverage of the skill acquisition. For the Gaze-Following skill, the OsL improved the time required to learn Gaze-Following. For the Looking-Back skill, both the XEP and OsL algorithms improved the time required to learn Looking-Back. The OsL seems to be effective on the time required to learn Gaze-Following and the time required to learn Looking-Back due to the synchronization problem described in section 3.1, while in the case of XEP, increasing the accuracy of the contingency evaluation, and as a result, the number of the acquired Looking-Backs seems to be the reason of the improvement. Thus, adopting both the algorithms will produce the best performance of the learning speed for the robot. The OsL uses weighted learning, which may increase the acquisition of the non-contingent skills, and the XEP may compensate it by increasing the accuracy of the contingency evaluation. However, we could not conclude anything because no significant effects of either of the algorithms and their interaction were found. Both the XEP and OsL improved the number of the expected transition. Therefore, using both of the algorithms are suggested to improve the predictability of the robot's behavior.

The most significant contribution of the current paper is building a real humanoid robot that could acquire complex social skills through sub-hour face-to-face interaction with a human while the previous work focused on the computer simulation or needed enormous interaction steps corresponding to several hours in the real world (Sumioka et al., 2010; Mugan and Kuipers, 2012; Mahzoon et al., 2016). It is worth noting that the proposed mechanism still succeeded in reproducing some infant developmental processes for social behavior resembling gaze following (Butterworth and Jarrett, 1991) and social referencing (Tomasello et al., 1995) as reported in the previous work (Sumioka et al., 2010; Mahzoon et al., 2016), although it is limited to involving the superficial similarities. Furthermore, it is also worth noting that the proposed mechanism could adapt to the behavioral changes in human, that is the emergence of a rewarding response to the behavioral changes in the robot, by extending the previously acquired skills. These features provide us with a research platform for further investigations of the flexible or variable developmental mechanism of human-like social skills in a dynamic and open-ended environment. However, since the current implementation was still limited to skills represented by combinations of several action and sensory variables, how to treat more rich variables for more complex skills will be the important future work.

4.5. Subjective Evaluation

4.5.1. Questionnaire and Result of Test

To evaluate whether the skill acquisition processes of the robot utilizing different algorithms make a difference in the subjective opinion of the participants about the quality of the interaction as well as the feeling about the intelligence of the robot, we conducted a subjective evaluation using a questionnaire. It consisted of seven questions, which were designated with Q1–Q7. The answers were proposed as five-level Likert scale, where 5 presented strongly agree and 1 presented strongly disagree. Additionally, to evaluate the transition of the answers over time, we administered the questionnaire every 200 steps, i.e., approximately every 10 min.

Figure 6 shows the average and standard deviation of the answers (described as score) to each question over time for each condition of the experiment. The statement used for each question is brought in the caption of the figure. A mixed-design three-way MANOVA was conducted with three independent variables (IVs) and seven dependent variables (DVs), to indicate the effect of using each algorithm (XEP and OsL) as two between subjects variables and also the course of time (hereafter denoted with "Time") as a within subjects variable on the score of the questions (score of each questionnaire Q1–Q7) as DVs of the test. The XEP and OsL indicated whether the corresponding algorithms were used in the experiment while the Time indicated the time that the questionnaire was taken and the score was obtained, which were consisted of four levels, i.e., 10, 20, 30, and 40 min. Also, a *post-hoc* power analysis was conducted to determine the observed power $(1-\beta)$ of the test, computed using $\alpha = 0.05$.

The result of the test suggested a significant multivariate effect of all three IVs, XEP (Wilk's $\Lambda = 0.677$, $F_{(7,38)} = 2.59, p = 0.027, \eta_p^2 = 0.323, 1-\beta = 0.826$), OsL (Wilk's $\Lambda = 0.635$, $F_{(7,38)} = 3.12, p = 0.011, \eta_p^2 = 0.365, 1-\beta = 0.899$) and Time (Wilk's $\Lambda = 0.179$, $F_{(21,24)} = 5.26, p < 0.001, \eta_p^2 = 0.821, 1-\beta = 1.000$) across the DVs. However, no significant interaction was revealed in the result of the multivariate test. In the follow-up univariate ANOVAs, while several main effects were revealed, no interaction between the factors was confirmed. The result of the test was summarized in **Table 5**, where only the revealed significance was mentioned. In this table, for the within subjects variable Time, except of Q3 and Q5, the result of Mauchly's test indicated that the assumption of sphericity had been violated, therefore the degrees of freedom were corrected using Greenhouse-Geisser estimates of sphericity. Note that the univariate ANOVAs were conducted using Bonferroni adjusted alpha levels of .007 concerning the number of the questions, i.e., .05/7. Also, to facilitate the discussion, the result of the univariate ANOVAs were summarized in the top left side of each graphs in **Figure 6**, indicating the p value of the main effect for the independent variables. As shown in the figure, it was revealed that the XEP algorithm was effective to increase the score of perceived intention (Q1), expected reaction (Q2), and human enjoyment (Q4), while the OsL algorithm was also effective to increase these scores in addition to the other twos; robot enjoyment (Q3) and robot's conformation (Q6). Also, this figure and **Table 5** showed that the variable Time had main effect on all DVs, except of Q7.

To indicate how the scores were changed in the course of time, the *post hoc* multiple comparison using Dunnett's method was conducted for the variable Time, using Bonferroni adjusted alpha levels of .007 concerning the number of the questions, i.e., .05/7. In this comparison, the score at Time=10min was compared with the others, i.e., Time = 20, 30, and 40min. The result of the comparison was summarized in **Table 6**. As shown in the table, for all of the questions mentioned in the table, the score was significantly increased from Time = 10 min to all of the other Times, except for one case, i.e.m, for Time = 20 min in Q5. In other words, it was revealed that compared to the first subjective evaluation (i.e., at Time = 10 min), the evaluation of the perceived intention (Q1), expected reaction (Q2), robot enjoyment (Q3), human enjoyment (Q4), and robot's conformation (Q6) were significantly increased after the second evaluation (i.e., at Time = 20 min), while the evaluation for robot's mind (Q5) was significantly increased after the third evaluation (i.e., at Time = 30 min).

4.5.2. Discussion

The factor of time was effective on the improvement of all of the question items, except for human conformation (Q7). The improvement of the scores from 10 to 20 or 30 min indicated that, in course of time, the robot even with neither of the proposed algorithms seemed to became looking more positive in many aspects, understanding human's intention (Q1), reacting as human expected (Q2), having its own mind (Q5), and conforming its behavior to human's behavior (Q6), and enjoying interaction (Q3) while human became enjoying interaction (Q4). This suggests that the basic developmental algorithm of the skill acquisition worked properly based on the subjective criteria.

The XEP and OsL algorithm were both effective on improving the score of some questions. Meanwhile, they improved the learning performance of skills necessary to follow the human's instruction, which is Gaze-Following and Looking-Back. Therefore, the human subjects seemed to feel that the robot understood his/her intention (Q1) of instruction, reacted as he/she expected (Q2), and consequently he/she could praise the robot more often which would make the interaction more enjoyable for the human (Q4). On the other hand, only the OsL had the main effect on the scores of robot's enjoyment (Q3) and robot conformation (Q6). It is considered to be sub-effects of the stopping behavior of the robot toward the human adopted in the OsL, which could represent the robot's attitude to positively follow the human's behavior. However, the results of the ANOVA for Q5 and Q7 had no significant effect of either of the proposed algorithms. A *post hoc* interview revealed that some subjects found negative meaning in the word "human conformation"

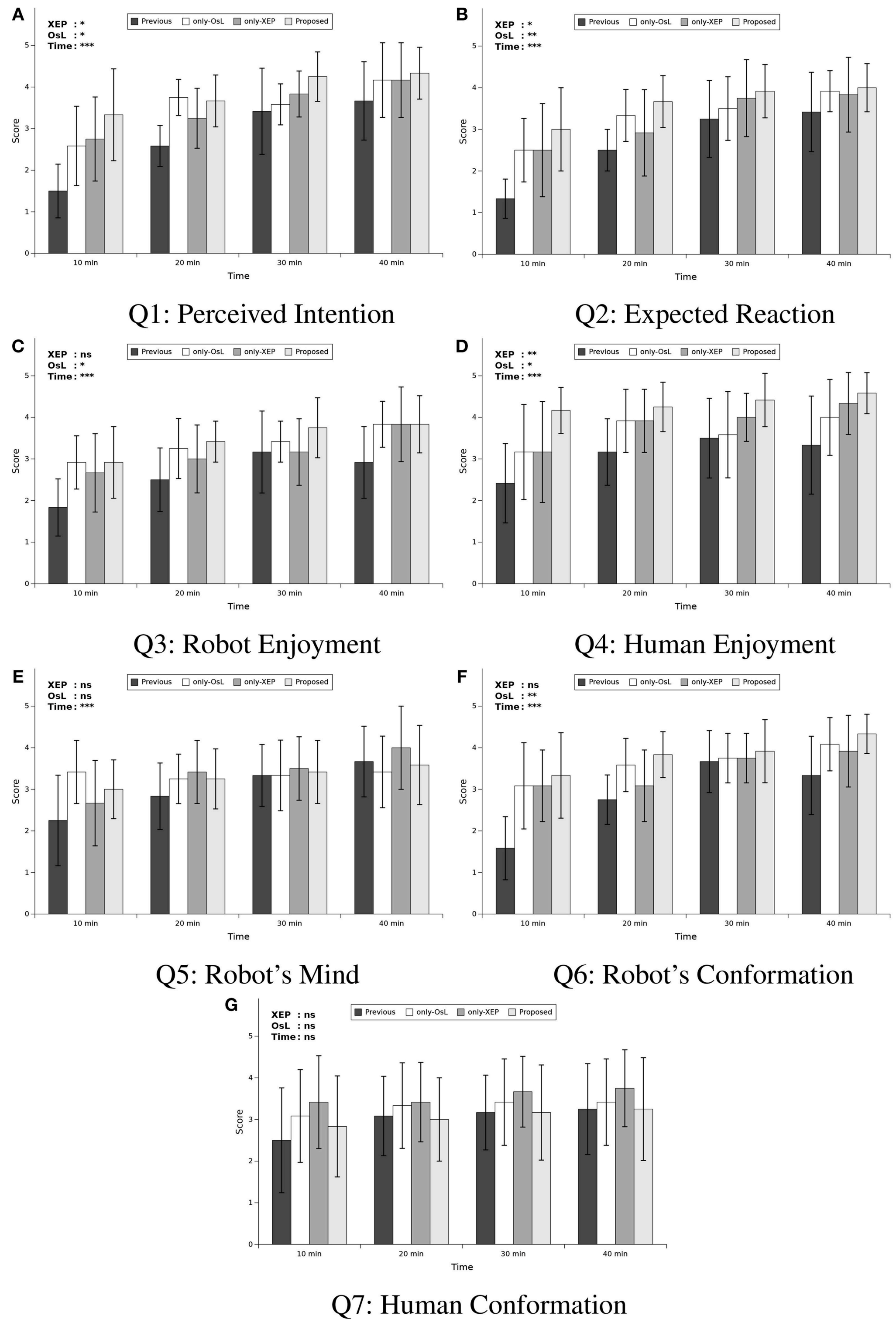


FIGURE 6 | Mean scores of questionnaire. **(A)** Q1: The robot understood my intention, **(B)** Q2: The robot reacted as I expected, **(C)** Q3: The robot looks like it is enjoying the interaction, **(D)** Q4: I enjoyed the interaction, **(E)** Q5: I felt that the robot had its own mind and behaved based on it, **(F)** Q6: The robot conformed its behavior to my behavior, and **(G)** Q7: I conformed my behavior to robot's behavior. Each sub-figure includes four comparisons in each time step ($t = 10$, 20, 30, and 40 min) among four conditions of learning method: previous work, learning only with Ostensive-cue sensitive Learning (only-OsL), learning only with Exclusive Evaluation of Policies (only-XEP), and learning both with the OsL and XEP (proposed). At the top left side of each graph, significant levels of main effects in the follow-up univariate ANOVA with Time as within-factor and OsL and XEP as between-subject factors are mentioned. The *p*-values are denoted by $^{***}p < 0.001/7$, $^{**}p < 0.01/7$, $^{*}p < 0.05/7$, and ns, not significant, considering Bonferroni correction concerning the number of the questions, i.e., 7 question items. Note that any interactions were not confirmed with the ANOVA.

TABLE 5 | Result of the follow-up univariate ANOVA for the questionnaire.

Item	Factor	df_1	df_2	$F(df_1, df_2)$	p	η_p^2	$1-\beta$
Q1	XEP	1	44	10.7	0.002	0.196	0.949
	OsL	1	44	11.5	0.001	0.208	0.963
	Time	2.01	88.5	47.8	0.000	0.521	1.000
Q2	XEP	1	44	9.41	0.004	0.176	0.917
	OsL	1	44	12.0	0.001	0.215	0.969
	Time	2.53	111	38.3	0.000	0.465	1.000
Q3	OsL	1	44	10.0	0.003	0.186	0.934
	Time	3	132	24.2	0.000	0.355	1.000
Q4	XEP	1	44	14.6	0.000	0.249	0.989
	OsL	1	44	7.96	0.007	0.153	0.862
	Time	2.32	102	12.1	0.000	0.215	1.000
Q5	Time	3	132	11.8	0.000	0.211	1.000
Q6	OsL	1	44	15.2	0.000	0.256	0.992
	Time	2.04	89.7	26.7	0.000	0.378	1.000

Only the significant factors are mentioned for each question items, considering Bonferroni adjusted alpha levels of .007 (i.e., .05/7) concerning the number of the questions, i.e., 7 questions. The degree of freedom for the factor and the error for the F-test was denoted with df_1 and df_2, respectively. The result of the F-test [$F(df_1, df_2)$], p-value (p), effect size (η_p^2) and the power of the test ($1-\beta$) were denoted as well.

TABLE 6 | Result of the multiple comparison with Dunnett's method for the variable Time considering Bonferroni adjusted alpha levels of 0.007 (i.e., 0.05/7) concerning the number of the questions, i.e., 7 questions.

Item	M_1	SD_1	T_2	M_2	SD_2	p	Cohen's d	$1-\beta$
Q1	2.54	1.17	20	3.31	0.75	0.000	0.787	1.000
			30	3.77	0.78	0.000	1.231	1.000
			40	4.08	0.90	0.000	1.482	1.000
Q2	2.33	1.08	20	3.10	0.86	0.000	0.792	1.000
			30	3.60	0.87	0.000	1.297	1.000
			40	3.79	0.80	0.000	1.529	1.000
Q3	2.58	0.92	20	3.04	0.80	0.001	0.532	0.950
			30	3.38	0.82	0.000	0.912	1.000
			40	3.60	0.87	0.000	1.140	1.000
Q4	3.22	1.19	20	3.81	0.84	0.000	0.566	0.970
			30	3.88	0.91	0.000	0.609	0.985
			40	4.06	1.00	0.000	0.759	0.999
Q5	2.83	1.02	20	3.19	0.76	0.051	0.389	0.752
			30	3.40	0.79	0.001	0.616	0.987
			40	3.67	0.95	0.000	0.845	1.000
Q6	2.77	1.17	20	3.31	0.80	0.001	0.540	0.956
			30	3.77	0.69	0.000	1.040	1.000
			40	3.92	0.85	0.000	1.122	1.000

In the comparison, the Time = 10 was compared with the others. In the columns of the table, the question item (Item), the mean and SD of the scores for the question at Time = 10 (M_1 and SD_1, respectively), the time that compared with (T_2), the mean and SD of the scores for T_2 (M_2 and SD_2, respectively), the p-value of the comparison (p), the effect size (Cohen's d) and the power of the test ($1-\beta$) were indicated.

(Q7). Also, the meaning of "mind" in Q5 might largely vary among the subjects. These might mean that they are difficult to be directly used as subjective measures.

In sum, we compared the result of the subjective evaluation of the participants in different conditions of the experiment related to their opinion about the quality of the interaction as well as the intelligence of the robot. The result showed a significant effect of the OsL and XEP algorithm on the evaluation. As described in section 1, when a caregiver recognizes a contingent and intelligent reply from an infant, he/she usually changes his/her behavior to teach a new concept to the infant. Assuming that the increase in the result of the evaluation expressing the higher level of such recognition, we can conclude that the proposed algorithms are significantly effective in inducing the caregiver to change his/her behavior and teach the infant robot a new concept. Consequently, the OsL and XEP could successfully contribute to an increase in an open-ended development of the infant robot. However, the items of the questionnaire applied in this part were not completely independent and there were correlation among some of them. Since a set of questionnaire to evaluate how the impression of the subjects about the robot was changed along with its development is not established yet, studying and inventing a suitable set with a factor analysis for such evaluation is an important future work of this field.

5. CONCLUSION

In this paper, we proposed two novel algorithms to improve the performance of the social skill learning of an infant robot during interaction with a human caregiver: namely the Ostensive-cue sensitive Learning (OsL) and the Exclusive Evaluation of Policies (XEP) algorithms. The OsL was inspired by the natural pedagogy of the human being and proposed a synchronized weighted learning mechanism based on the ostensive signals of the caregiver. The XEP algorithm proposed a way to improve the accuracy of the contingency evaluation by separating the histogram of the contingencies related to the acquired policies and atomic variables. The OsL was expected to increase the learning speed of the robot, while the XEP was expected to improve the accuracy of the contingency evaluation, especially those related to the acquired policies (i.e., complex skills).

The results of our humanoid robot experiment with human subjects showed that the OsL was effective in increasing the learning speed of the simple and complex skills, and consequently increasing the number of learned skills by the robot; while the XEP increased the accuracy of the contingency evaluation and was effective in increasing the coverage of complex skills as well as the time-steps required for the learning. These improvements resulted in enabling the infant robot and the human subject to predict each others' behavior. As a result, statistical analysis of the experiment showed a significant effect of both algorithms on increasing the number of the expected transition of the infant robot, the subjective evaluation of the human participants about the quality of the interaction and the intelligence of the robot. Since the level of the recognition of the human caregiver about the intelligence of the robot has an impact on the teaching tendency of the caregiver, the increase in the subjective evaluation can be expressed as a contribution of the proposed algorithms on increasing the opportunity of the open-ended development of the infant robot. Finally, the proposed mechanism of this paper enabled the robot to learn some primitive social skills *within a short time-step of a real-world interaction* with a human subject: simple skills such as the Gaze-Following behavior after 8 min, and complex skills such as Looking-Back behavior after 25 min.

However, the variables utilized in this work were assumed to be quantized, and the modality of the sensory and action variables of the robot were still few. Utilizing dynamic quantization methods such as that proposed in the previous work (Mugan and Kuipers, 2012) could be a way to treat continuous variables. Meanwhile, the way to dynamically adapt the learning parameters of the system to the developmental change in quantization level would be an important topic. Research on this topic will propose an insight about the developmental models, which may be compared with the model of human infant. Moreover, adding more modalities to the variables, such as the voice of the caregiver to the sensory variables, and speaking/uttering ability to the action variables of the robot could increase the complexity of the interaction as well as that of acquired skills by the robot. Nevertheless, treating with the probable huge varieties of the caregiver's behavior will be one of the challenging issues for the implementation of the developmental robot in such an environment. These problems are needed to be considered as the main topics of the future work.

AUTHOR CONTRIBUTIONS

HM wrote computer code, performed the modeling and subject experiment, analyzed output data, and wrote the paper manuscript. YY supervised the modeling, experimental setup, subject experiment, and data analyses. HI supervised the project.

FUNDING

The work reported in this paper has been supported by Grant-in-Aid for Scientific Research on Innovative Areas Constructive Developmental Science (No.24119003) from The Ministry of Education, Culture, Sports, Science, and Technology, as well as ISHIGURO Symbiotic Human-Robot Interaction Project (grant number JPMJER1401) from the JST ERATO, Japan.

REFERENCES

Adamson, L. B. (1995). *Communication Development During Infancy.* Madison, WI: Brown & Benchmark.

Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., et al. (2009). Cognitive developmental robotics: a survey. *IEEE Trans. Auton. Mental Dev.* 1, 12–34. doi: 10.1109/TAMD.2009.2021702

Barto, A. G. (2013). "Intrinsic motivation and reinforcement learning," in *Intrinsically Motivated Learning in Natural and Artificial Systems,* eds G. Baldassarre and M. Mirolli (Berlin: Springer), 17–47.

Breazeal, C. (2004). Social interactions in HRI: the robot view. *IEEE Trans. Syst. Man Cybern. Part C* 34, 181–186. doi: 10.1109/TSMCC.2004.826268

Butterworth, G., and Jarrett, N. (1991). What minds have in common is space: spatial mechanisms serving joint visual attention in infancy. *Br. J. Dev. Psychol.* 9, 55–72. doi: 10.1111/j.2044-835X.1991.tb00862.x

Corkum, V., and Moore, C. (1995). *Development of Joint Visual Attention in infants*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Csibra, G., and Gergely, G. (2009). Natural pedagogy. *Trends Cogn. Sci.* 13, 148–153. doi: 10.1016/j.tics.2009.01.005

Csibra, G., and Gergely, G. (2011). Natural pedagogy as evolutionary adaptation. *Philos. Trans. R. Soc. B Biol. Sci.* 366, 1149–1157. doi: 10.1098/rstb.2010.0319

Degris, T., Sigaud, O., and Wuillemin, P.-H. (2006). "Learning the structure of factored markov decision processes in reinforcement learning problems," in *Proceedings of the 23rd International Conference on Machine Learning* (Pittsburgh, PA: ACM), 257–264.

Imai, M., Ono, T., and Ishiguro, H. (2003). Physical relation and expression: joint attention for human-robot interaction. *IEEE Trans. Indus. Electron.* 50, 636–643. doi: 10.1109/TIE.2003.814769

Jonsson, A., and Barto, A. (2007). "Active learning of dynamic bayesian networks in markov decision processes," in *International Symposium on Abstraction, Reformulation, and Approximation* (Whistler, BC: Springer), 273–284.

Kanda, T., Ishiguro, H., Imai, M., and Ono, T. (2004). Development and evaluation of interactive humanoid robots. *Proc. IEEE* 92, 1839–1850. doi: 10.1109/JPROC.2004.835359

Kaplan, F., and Hafner, V. V. (2006). The challenges of joint attention. *Interact. Stud.* 7, 135–169. doi: 10.1075/is.7.2.04kap

Mahzoon, H., Yoshikawa, Y., and Ishiguro, H. (2016). Social skill acquisition model through face-to-face interaction: local contingency for open-ended development. *Front. Robot. AI* 3:10. doi: 10.3389/frobt.2016.00010

Moore, C., and Dunham, P. (2014). *Joint Attention: Its Origins and Role in Development*. New York, NY: Psychology Press.

Movellan, J. R., and Watson, J. S. (2002). "The development of gaze following as a bayesian systems identification problem," in *Proceedings of the 2nd International Conference on Development and Learning* (Los Alamitos, CA), 34–40.

Mugan, J., and Kuipers, B. (2012). Autonomous learning of high-level states and actions in continuous environments. *IEEE Trans. Auton. Mental Dev.* 4, 70–86. doi: 10.1109/TAMD.2011.2160943

Mundy, P., Card, J., and Fox, N. (2000). EEG correlates of the development of infant joint attention skills. *Dev. Psychobiol.* 36:325. doi: 10.1002/(SICI)1098-2302(200005)36:4<325::AID-DEV7>3.0.CO;2-F

Nagai, Y., Hosoda, K., Morita, A., and Asada, M. (2003). A constructive model for the development of joint attention. *Connect. Sci.* 15, 211–229. doi: 10.1080/09540090310001655101

Nehmzow, U., Gatsoulis, Y., Kerr, E., Condell, J., Siddique, N., and McGuinnity, T. M. (2013). "Novelty detection as an intrinsic motivation for cumulative learning robots," in *Intrinsically Motivated Learning in Natural and Artificial Systems,* eds G. Baldassarre and M. Mirolli (Berlin: Springer), 185–207.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evol. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Scaife, M., and Bruner, J. S. (1975). The capacity for joint visual attention in the infant. *Nature* 253, 265–266.

Schreiber, T. (2000). Measuring information transfer. *Phys. Rev. Lett.* 85:461. doi: 10.1103/PhysRevLett.85.461

Sumioka, H., Yoshikawa, Y., and Asada, M. (2010). Reproducing interaction contingency toward open-ended development of social actions: case study on joint attention. *IEEE Trans. Auton. Mental Dev.* 2, 40–50. doi: 10.1109/TAMD.2010.2042167

Sumioka, H., Yoshikawa, Y., Morizono, M., and Asada, M. (2013). "Socially developmental robot based on self-induced contingency with multi latencies," in *Advances in Cognitive Neurodynamics (III),* ed Y. Yamaguchi (Dordrecht: Springer), 251–258.

Tomasello, M. (1995). "Joint attention as social cognition," in *Joint Attention: Its Origins and Role in Development,* eds C. Moore and P. J. Dunham (Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.), 103–130.

Tomasello, M. (2009). *The Cultural Origins of Human Cognition*. Cambridge, MA: Harvard University Press.

Triesch, J., Teuscher, C., Deák, G. O., and Carlson, E. (2006). Gaze following: why (not) learn it? *Dev. Sci.* 9, 125–147. doi: 10.1111/j.1467-7687.2006.00470.x

Watson, J. S. (1972). Smiling, cooing, and "the game." *Merrill-Palmer Q. Behav. Dev.* 18, 323–339.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Artificial Development by Reinforcement Learning Can Benefit From Multiple Motivations

Günther Palm and Friedhelm Schwenker*

Institute of Neural Information Processing, Ulm University, Ulm, Germany

Research on artificial development, reinforcement learning, and intrinsic motivations like curiosity could profit from the recently developed framework of multi-objective reinforcement learning. The combination of these ideas may lead to more realistic artificial models for life-long learning and goal directed behavior in animals and humans.

Keywords: reinforcement learning, multi-objective, actor-critic design, artificial curiosity, artificial cognition, intrinsic motivation

OPEN ACCESS

Edited by:
Gianluca Baldassarre,
Italian National Research Council (CNR), Italy

Reviewed by:
Eiji Uchibe,
Advanced Telecommunications Research Institute International (ATR), Japan
Jaan Aru,
University of Tartu, Estonia

****Correspondence:***
Günther Palm
guenther.palm@uni-ulm.de

Specialty section:
This article was submitted to Computational Intelligence, a section of the journal Frontiers in Robotics and AI

Received: *30 May 2018*
Accepted: *16 January 2019*
Published: *14 February 2019*

Citation:
Palm G and Schwenker F (2019) Artificial Development by Reinforcement Learning Can Benefit From Multiple Motivations. Front. Robot. AI 6:6. doi: 10.3389/frobt.2019.00006

INTRODUCTION

Reinforcement learning (RL) is a well-established learning paradigm, first consolidated in the book of Sutton and Barto (1998) after the early years of artificial neural networks and machine learning, with strong roots in the mathematics of dynamical programming (Bellman, 1957) and in the early behavioral psychology of Pavlovian conditioning and learning (Rescorla and Wagner, 1972).

In recent years, plausible neural mechanisms for all essential components of RL have been found in the brain, in particular in the basal ganglia, but also in frontal cortical areas, perhaps involved in different versions of RL (Wiering and van Otterlo, 2012), which have been developed not only from a technical, but also from a neuroscientific motivation; overviews are given in Farries and Fairhall (2007), Botvinick et al. (2009), Chater (2009), Maia (2009), Joiner et al. (2017), and Wikenheiser and Schoenbaum (2016).

Also in recent developments of robotics, artificial agents, or artificial life, in particular when the focus is on learning interesting "cognitive" abilities or behaviors or on child-like "artificial development" (Oudeyer et al., 2007), the framework of RL is often used. If it is understood to include its continuous version, actor critic design (Bertsekas and Tsitsiklis, 1996; Prokhorov and Wunsch, 1997) reinforcement learning is a very general approach encompassing applications from Go-playing (Silver et al., 2016) to motor control (Miller et al., 1995; Kretchmara et al., 2001; Todorov, 2004; Schaal and Schweighofer, 2005; Lendaris, 2009; Riedmiller et al., 2009; Wong and Lee, 2010; Little and Sommer, 2011).

Here we are considering RL in the context of robotics or rather of artificial agents that learn to act appropriately in a simulated or real environment. Most often this involves continuous state and action spaces which cannot simply be discretized (Lillicrap et al., 2015). So usually the RL paradigm is combined with a neural network approach to represent the reward predicting function (Sutton and Barto, 1998; Oubbati et al., 2012, 2014; Faußer and Schwenker, 2015).

In this context there are a number of issues that this framework cannot easily accommodate:

1. the learning of several partially incompatible behaviors,
2. the balance between exploration and exploitation,
3. the development and integration of "meta-heurictics" like "curiosity" or "cautiousness,"
4. the problem of finding a "state space" and its partial observability,
5. the simulation of apparently changing strategies in animal behavior.

In reaction to the first issue one might argue that RL is just for one particular behavior, not for the combination of several behaviors; for this one would need to combine several instances of RL. Of course, one could also argue that each animal has just one behavior which maximizes its chance of survival and apparent particular behaviors or motives driving it must be subordinate to this ultimate goal, similarly in economic decision making the ultimate goal is financial utility (money) and it would be irrational to follow other rewards from time to time (as in the fairy tale of *Hans im Gück*). All this has been debated at length (e.g., Simon, 1955, 1991; Tisdell, 1996; Gigerenzer and Selten, 2002; Kahneman, 2003; Dayan and Niv, 2008; Dayan and Seymour, 2009; Glimcher et al., 2009; Chiew and Braver, 2011) leading to considerable doubts in a simple utilitarian view in economy and practically to various approaches extending basic RL, often in a hierarchical fashion (Barto et al., 2004; Botvinick et al., 2009). Even a human or robot Go-player has not only to consider Go strategies, but also (on a lower level) to control his arm movements when taking and placing a piece.

The balance between exploration and exploitation has been widely discussed in classical RL and even before that (e.g., Feldbaum, 1965). It has lead to various, often stochastic, amendments to the original basic method (Wiering and van Otterlo, 2012) without a convincing general solution that works well in most applications. This problem has also inspired more general approaches in more complex scenarios which add special "meta-objectives" like "curiosity" or "cautiousness" to the RL scheme (perhaps first by Schmidhuber, 1991), which again points toward a multi-objective approach. Recently these ideas are discussed in particular in the context of autonomous "cognitive" agents and their "artificial development" (Weng et al., 2001; Lungarella et al., 2003; Barto et al., 2004; Oudeyer et al., 2007).

In biology and human psychology or sociology it is clear that the state space (i.e., the total relevant state of the world) is far from being observable by the senses of the individual animal or human. It might even be doubted whether there is such a state at all. At least it is often asking too much to assume that the individual possesses a representation of the set or space of all possible states. Such scenarios are even outside the usual relatively broad POMDP (partially observable Markov decision process, see Kaelbling et al., 1996) formalism, so biologically motivated realizations of RL often rest on relatively simple versions of RL that don't require knowledge of a "state" in the sense of physics, but just rely on sensory and reward input.

Also the last issue is clearly at variance with the basic model of classical RL. However, when we consider the creation of artificial autonomous agents or artificial animals an obvious potential answer to all of these issues comes to mind: Such an agent or animal usually has several different, sometimes conflicting goals or motivations (e.g., food, drink, and sex) which cannot simply be combined linearly to form one general objective (Liu et al., 2015).

It therefore seems natural to use different instances of RL on different simplified state spaces, which contain incomplete information on different aspects of the physical state of the world, with different objectives or reward functions in different contexts or situations and somehow select the most important ones to determine the agent's behavior in each concrete situation. This means that one has to consider multiple objectives and their interaction in decision making. This problem is studied by a growing research community under the heading of "multiple objective reinforcement learning" (MORL).

The framework of MORL can be used to address and alleviate the 5 problems mentioned above. In fact, it is directly motivated from problems 1 and 5. The dilemma between exploration and exploitation (problem 2) is greatly alleviated by the simple observation that behavior guided by exploitation of one objective usually can be considered as exploration for all other objectives. The development of meta-heuristics or "intrinsic motivations" (issue 3) can be very useful also in technical applications; for the MORL framework advocated here the point is simply to put intrinsic motivations like curiosity or cautiousness side-by-side with the basic "extrinsic" motivation(s). Concerning the state-space (problem 4), in many practical applications a real "state-space" is unknown or at best partially observable. In this case the best one can do is to obtain a sufficiently rich approximate representation for it based on sensory data and reinforcement signals, and more such signals are certainly better than less for this purpose.

REPRESENTING THE STATE SPACE

In order to obtain an approximate state representation by learning from experience, one can use a neural network, typically a multilayer perceptron (MLP) or "deep network" or methods of reservoir computing (Maass et al., 2002; Jaeger and Haas, 2004) for continuous temporal dynamics, or a combination of both. In complex control problems (Koprinkova-Hristova and Palm, 2010) such a representation is often called a "forward model." So the agent (biological or artificial) tries to learn a "state representation network," i.e., a (typically recurrent) network that predicts the next state from a representation of the current state, which integrates sensory input information over time and can be used as input to the evaluation or critic network in the usual situation where the current sensory input is insufficient to determine the "state" of the environment; see for example (Sutton and Barto, 1981; Schmidhuber, 1991; Dayan and Sejnowski, 1996; Herrmann et al., 2000; Gläscher et al., 2010). Such a network can be used as the basis for a second network representing the quality or value function in reinforcement learning or actor-critic design.

The use of neural networks or parameterized approximators as estimators of the state-value or state-action-value function is a way to deal with large or continuous action and state spaces. The approximating function may be a linear or nonlinear function of their parameters, but linear approximators show limitations in their expressive power, while convergence of learning is quaranteed. Nonlinear approximators, typically neural networks, are universal approximators (Cybenko, 1989), but often show instable behavior during learning. During the last years increasingly complex networks are used in RL for large and continuous state spaces; in addition to classical multilayer perceptrons or radial basis function networks, also trainable recurrent neural networks (Hagenbuchner et al., 2017) or echo-state-networks (Scherer et al., 2008; Oubbati et al., 2012, 2013, 2014; Koprinkova-Hristova et al., 2013) are used, and particular methods have been developed to improve the stability of learning (Hafner and Riedmiller, 2011; Silver et al., 2014; Faußer and Schwenker, 2015; Lillicrap et al., 2015; Parisi et al., 2017). Recently, deep neural networks such as autoencoders and convolutional neural networks have been applied for representation learning and used in combination with RL methods to learn complex decision task from raw data (Lillicrap et al., 2015; Mnih et al., 2015; Mossalam et al., 2016; Srinivasan et al., 2018).

In any case it is practically important for MORL to use one and the same network as a basis to create a sufficiently rich representation in order to train all different objectives (critics and actors) as outputs of the last layer (Mossalam et al., 2016).

Based on the sensory input alone, but also on such an approximate state representation, it often will not be possible to predict the expected reward or the next state with certainty. In a neural network for classification, for example, this uncertainty will be expressed by submaximal activation of several output neurons and these activations may be interpreted as *a posteriori* probabilities of the various outcomes (states or values); the uncertainty in estimating the expected reward is often measured by its variance. Beyond variance, there are various formalisms for calculating measures of certainty or uncertainty from these probabilities, often in terms of information theory (Palm, 2012), and several approaches to incorporate measures of uncertainty, or of "novelty" or "surprise" into the choice of appropriate actions in reinforcement learning (e.g., MacKay, 1992; Sporns and Pegors, 2003; Little and Sommer, 2011; Tishby and Polani, 2011; Sledge and Príncipe, 2017); much of this is reviewed and discussed by Schmidhuber (1997) or Schmidhuber (2003) also in relation to the exploration-exploitation dilemma (Dayan and Sejnowski, 1996; Auer, 2002; Tokic and Palm, 2012; Tokic et al., 2013). Again these practically important considerations point toward MORL, for example in the direction of additional "meta-objectives" like curiosity or cautiousness (Wiering and Schmidhuber, 1998; Uchibe and Doya, 2008; Oubbati et al., 2013). It is often useful to consider at least two versions of the primary objective, namely its expected value and an estimate of the value that can at least be obtained with a reasonably high probability (e.g., the 5-percentile).

The MORL idea transforms the original problem of learning one behavior that is useful in all circumstances into a problem of designing an appropriate architecture for learning and decision making that combines several (probably hierarchically organized) instances or stages of classical RL and possibly other methods of learning or decision making (Oubbati and Palm, 2010).

MULTI-OBJECTIVE REINFORCEMENT LEARNING

A framework for studying these problems in the restricted realm of reinforcement learning, which has recently gained increasing popularity, is called MORL (see Roijers et al., 2013; Liu et al., 2015). We would like to propose to use this framework as a starting point to tackle the broader architectural problem in some concrete scenarios, which occur quite naturally in many technical optimization and control problems and have been elaborated in the MORL community, some examples (Deep Sea Treasure, Bonas World, Cart Pole, Water Reservoir, Resource Gathering, Predator Prey) are described in Drugan et al. (2017) and the literature cited therein; see also Vamplew et al. (2011).

The difference of MORL to classical RL is quite simple: If we think in terms of actor-critic design, where essentially an evaluation of the agent's actions is learned in a POMDP and where this evaluation function may be learned by a neural network, now we just have a vector of evaluations instead of a single value (in the output layer of the network). Similarly there is now an actor for each component of the evaluation vector suggesting an appropriate action for that particular value, objective, or motive. This model clearly leads to the problem how to combine the different objectives and suggested actions in order to decide on the next action. This problem has been discussed thoroughly in the MORL community; for an overview see Liu et al. (2015) and Drugan et al. (2017) and we will contribute a few ideas on this issue in terms of the computational architecture. The most common idea is to combine the different reward values into a weighted sum and take the best action for this combination. More complex methods consider the so-called *pareto-front*, well-known from classical multi-objective optimization. In fact, much of the discussion on optimal decision making for multiple objectives and methods for finding the pareto-optimal solutions (Das and Dennis, 1998; Miettinen, 1999; Mueller-Gritschneder et al., 2009; Motta et al., 2012) can be useful for MORL (see Van Moffaert and Nowé, 2014; Pirotta et al., 2015; Vamplew et al., 2017).

Once the most appropriate action has been determined and carried out, each of the actors and critics is able to learn something from its outcome leading to a modification of the corresponding neural networks, usually through backpropagation of the expected reward update or temporal difference.

From introspection, but also from behavioral animal experiments one gets the impression that each of these motives enters the final evaluation and decision with its own weight or "urgency" that may vary with time, depending on the agent's needs, which implies that there is no fixed "trading relation" between the different motives and their corresponding reward values, so they cannot be reduced to just one value. Modeling artificial agents in this wider framework entails some new problems and tasks, which may also lead to new interesting research projects and interactions with behavioral biologists and psychologists.

Here we describe the basic theoretical framework for this approach:

1. Given n motives, n current predicted values $(v_1, \ldots, v_n)$, and n "urgency weights" $(w_1, \ldots, w_n)$ for them, how do we combine them to one value that should be maximized by the next action? There are different more or less obvious ideas for this (see e.g., Boutilier, 2002; Castelletti et al., 2002; Natarajan and Tadepalli, 2005; Wiering and De Jong, 2007)

also motivated by modeling animal behavior, or reflecting the introspective difference between positive and negative rewards, or between goal seeking and pain avoidance, the most obvious and simple being the weighted sum $v = \sum_i w_i v_i$. At the opposite extreme we would follow the one objective that has maximal $w_i v_i$, or we could consider a minimal value for some objectives as a constraint in maximizing the weighted sum of the others. Here the "higher" motives like curiosity are put side-by-side with "lower" ones like "hunger," which may be psychologically somewhat unsettling, but might actually work. We first encountered this idea in the work of Dörner (2001), see also Bach (2009) and Bach (2012).

2. For each of the motives, in addition to defining the corresponding rewards r_i we have to model their "urgency function" $w_i(t)$. This may involve a dynamical system model of the agent's body and as such may be considered as part of the world model. In particular, it will use the corresponding rewards $r_i(t)$ as inputs. In extreme cases w_i may even be constant or it may simply integrate the incoming rewards as

$$\dot{w}_i(t) = a - br_i(t) \quad \text{or} \quad \tau \dot{w}_i(t) = -w_i(t) - br_i(t) + a$$

but much more is easily conceivable, for instance involving thresholds at which the urgency changes drastically. The development of such dynamical models of urgency may be an interesting line of research also in modeling animal behavior. Actually, the simple integration model was probably first introduced informally by Lorenz (1978).

3. It is now possible to introduce some more "cognitive" motives like "curiosity" (see also Pisula, 2009), for which we have to define $r_i(t)$ and $w_i(t)$. For example for curiosity it is natural to define surprising events as rewarding, where surprise may be defined as $-\log p$ relative to a probabilistic world model that the agent may have learnt (Palm, 2012). More concretely, if in world state x the agent receives the observation $o(x)$, or the state description $d(x)$ (Palm, 2012), which has the probability $p(x) = p(d(x))$ in his current model, then his surprise is $-\log p(x)$. Then again $w_i(t)$ can be defined for example by an integration model.

4. Finally we have to decide for the optimal action. Given our estimates for the temporal rewards and urgencies of the different motives and also our momentary combined reward, we can use methods of multi-objective or of plain optimization to find the optimal action. As a starting point we can use the actor outputs for the individual motives and perhaps try their combinations. Practical methods for finding a reasonable solution to the optimization problem in short time are also discussed in the literature on RL and MORL (Handa, 2009; Kooijman et al., 2015; Brys et al., 2017; Parisi et al., 2017; Vamplew et al., 2017).

This leads to an extended RL-architecture, which may be biologically more realistic. Such a more complex architecture also offers interesting additional possibilities for improving behaviors by learning: The existence of more objectives compared to just one, generates a richer representation of (the value of) the current situation, which can be used also to improve the sensory-based world model. It also gives a new perspective on the exploration-exploitation dilemma, since following exploitation of one objective may serve as exploration of the others. We have presented a basic layout of such a multi-objective agent architecture and started some preliminary experiments on it (Oubbati et al., 2013, 2014), but we believe that much more can and should be done in this direction.

AUTHOR CONTRIBUTIONS

GP: involved in preparing the concept of the paper, and writing of the paper; FS: writing of paper including literature work and proofreading.

REFERENCES

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.* 3, 397–422.

Bach, J. (2009). *Principles of Synthetic Intelligence*. New York, NY: Oxford University Press.

Bach, J. (2012). A framework for emergent emotions, based on motivation and cognitive modulators. *Int. J. Synthet. Emot.* 3, 43–63. doi: 10.4018/jse.2012010104

Barto, A. G., Singh, S., and Chentanez, N. (2004). "Intrinsically motivated learning of hierarchical collections of skills," in *Proceedings of the 3rd International Conference on Development and Learning* (Cambridge, MA), 112–119.

Bellman, R. E. (1957). *Dynamic Programming*. New York, NY: Dover Publications, Incorporated.

Bertsekas, D. P., and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming, 1st Edn.* Belmont, MA: Athena Scientific.

Botvinick, M. M., Niv, Y., and Barto, A. C. (2009). Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition* 113, 262–280. doi: 10.1016/j.cognition.2008.08.011

Boutilier, C. (2002). "A pomdp formulation of preference elicitation problems," in *AAAI/IAAI* (Edmonton, AB), 239–246.

Brys, T., Harutyunyan, A., Vrancx, P., Nowé, A., and Taylor, M. E. (2017). Multi-objectivization and ensembles of shapings in reinforcement learning. *Neurocomputing* 263, 48–59. doi: 10.1016/j.neucom.2017.02.096

Castelletti, A., Corani, G., Rizzoli, A., Soncini Sessa, R., and Weber, E. (2002). "Reinforcement learning in the operational management of a water system," in *IFAC Workshop on Modeling and Control in Environmental Issues* (Yokohama), 303–308.

Chater, N. (2009). Rational and mechanistic perspectives on reinforcement learning. *Cognition* 113, 350–364. doi: 10.1016/j.cognition.2008.06.014

Chiew, K. S., and Braver, T. S. (2011). Positive affect versus reward: emotional and motivational influences on cognitive control. *Front. Psychol.* 2:279. doi: 10.3389/fpsyg.2011.00279

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Cont. Signals Syst.* 2, 303–314. doi: 10.1007/BF02551274

Das, I., and Dennis, J. E. (1998). Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM J. Optim.* 8, 631–657. doi: 10.1137/S1052623496307510

Dayan, P., and Niv, Y. (2008). Reinforcement learning: the good, the bad and the ugly. *Curr. Opin. Neurobiol.* 18, 185–196. doi: 10.1016/j.conb.2008.08.003

Dayan, P., and Sejnowski, T. J. (1996). Exploration bonuses and dual control. *Mach. Learn.* 25, 5–22. doi: 10.1007/BF00115298

Dayan, P., and Seymour, B. (2009). "Values and actions in aversion," in *Neuroeconomics: Decision Making and the Brain*, eds P. W. Glimcher, C. F.

Camerer, E. Fehr, and R. A. Poldrack (San Diego, CA: Elsevier Academic Press), 175–191.

Dörner, D. (2001). *Bauplan für eine Seele*. Reinbek: Rowohlt.

Drugan, M. M., Wiering, M., Vamplew, P., and Chetty, M. (2017). Special issue on multi-objective reinforcement learning. *Neurocomputing* 263, 1–2. doi: 10.1016/j.neucom.2017.06.020

Farries, M. A. and Fairhall, A. L. (2007). Reinforcement learning with modulated spike timing–dependent synaptic plasticity. *J. Neurophysiol.* 98, 3648–3665. doi: 10.1152/jn.00364.2007

Faußer, S., and Schwenker, F. (2015). Neural network ensembles in reinforcement learning. *Neural Process. Lett.* 41, 55–69. doi: 10.1007/s11063-013-9334-5

Feldbaum, A. (1965). *Optimal Control Systems*. New York, NY: Academic Press.

Gigerenzer, G., and Selten, R. (2002). *Bounded Rationality: The Adaptive Toolbox*. Cambridge: MIT Press.

Gläscher, J., Daw, N., Dayan, P., and O'Doherty, J. P. (2010). States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron* 66, 585–595. doi: 10.1016/j.neuron.2010.04.016

Glimcher, P. W., Camerer, C. F., Fehr, E., and Poldrack, R. A. (2009). *Introduction: A Brief History of Neuroeconomics*. San Diego, CA: Elsevier Academic Press.

Hafner, R., and Riedmiller, M. (2011). Reinforcement learning in feedback control. *Mach. Learn.* 84, 137–169. doi: 10.1007/s10994-011-5235-x

Hagenbuchner, M., Tsoi, A. C., Scarselli, F., and Zhang, S. (2017). "A fully recursive perceptron network architecture," in *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017* (Honolulu, HI) , 1–8.

Handa, H. (2009). "Solving multi-objective reinforcement learning problems by eda-rl-acquisition of various strategies," in *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on* (Pisa: IEEE), 426–431.

Herrmann, J. M., Pawelzik, K., and Geisel, T. (2000). Learning predictive representations. *Neurocomputing* 32–33, 785–791. doi: 10.1016/S0925-2312(00)00245-9

Jaeger, H., and Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* 304, 78–80. doi: 10.1126/science.1091277

Joiner, J., Piva, M., Turrin, C., and Chang, S. W. (2017). Social learning through prediction error in the brain. *npj Sci. Learn.* 2:8. doi: 10.1038/s41539-017-0009-2

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: a survey. *J. Artif. Intell. Res.* 4, 237–285. doi: 10.1613/jair.301

Kahneman, D. (2003). Maps of bounded rationality: psychology for behavioral economics. *Am. Econ. Rev.* 93, 1449–1475. doi: 10.1257/000282803322655392

Kooijman, C., de Waard, M., Inja, M., Roijers, D. M., and Whiteson, S. (2015). Pareto local policy search for momdp planning. *22th ESANN* (Bruges), 53–58. Available online at: http://www.i6doc.com/en/

Koprinkova-Hristova, P., Oubbati, M., and Palm, G. (2013). Heuristic dynamic programming using echo state network as online trainable adaptive critic. *Int. J. Adapt. Control Signal Process.* 27, 902–914. doi: 10.1002/acs.2364

Koprinkova-Hristova, P., and Palm, G. (2010). "Adaptive critic design with esn critic for bioprocess optimization," in *International Conference on Artificial Neural Networks* (Berlin; Heidelberg: Springer), 438–447.

Kretchmara, R. M., Young, P. M., Anderson, C. W., Hittle, D. C., Anderson, M. L., and Delnero, C. (2001). "Robust reinforcement learning control," in *American Control Conference, 2001. Proceedings of the 2001*, Vol. 2 (Arlington, VA: IEEE), 902–907.

Lendaris, G. G. (2009). "A retrospective on adaptive dynamic programming for control," in *Proceedings of the 2009 International Joint Conference on Neural Networks, IJCNN'09* (Piscataway, NJ: IEEE Press), 945–952.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. *arXiv[preprint]. arXiv:1509.02971*.

Little, D. Y., and Sommer, F. T. (2011). Learning in embodied action-perception loops through exploration. *arXiv[preprint]. arXiv:1112.1125*.

Liu, C., Xu, X., and Hu, D. (2015). Multiobjective reinforcement learning: a comprehensive overview. *IEEE Trans. Syst. Man Cybern. Syst.* 45, 385–398. doi: 10.1109/TSMC.2014.2358639

Lorenz, K. (1978). *Vergleichende Verhaltensforschung: Grundlagen der Ethologie*. New York, NY: Springer.

Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *Connect. Sci.* 15, 151–190. doi: 10.1080/09540090310001655110

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* 14, 2531–2560. doi: 10.1162/089976602760407955

MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural Comput.* 4, 590–604. doi: 10.1162/neco.1992.4.4.590

Maia, T. V. (2009). Reinforcement learning, conditioning, and the brain: successes and challenges. *Cogn. Affect. Behav. Neurosci.* 9, 343–364. doi: 10.3758/CABN.9.4.343

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. New York, NY: Springer.

Miller, W. T., Sutton, R. S., and Werbos, P. J. (eds.). (1995). *Neural Network and Control*. Cambridge MA: MIT Press.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518:529. doi: 10.1038/nature14236

Mossalam, H., Assael, Y. M., Roijers, D. M., and Whiteson, S. (2016). Multi-objective deep reinforcement learning. *arXiv[preprint]. arXiv:1610.02707*.

Motta, R. S., Afonso, S. M. B., and Lyra, P. R. M. (2012). A modified nbi and nc method for the solution of n-multiobjective optimization problems. *Struct. Multidiscip. Optim.* 46, 239–259. doi: 10.1007/s00158-011-0729-5

Mueller-Gritschneder, D., Graeb, H., and Schlichtmann, U. (2009). A successive approach to compute the bounded Pareto front of practical multiobjective optimization problems. *SIAM J. Optim.* 20, 915–934. doi: 10.1137/080729013

Natarajan, S., and Tadepalli, P. (2005). "Dynamic preferences in multi-criteria reinforcement learning," in *Proceedings of the 22nd International Conference on Machine Learning* (Bonn: ACM), 601–608.

Oubbati, M., Kord, B., Koprinkova-Hristova, P., and Palm, G. (2014). Learning of embodied interaction dynamics with recurrent neural networks: some exploratory experiments. *J. Neural Eng.* 11:026019. doi: 10.1088/1741-2560/11/2/026019

Oubbati, M., Oess, T., Fischer, C., and Palm, G. (2013). "Multiobjective reinforcement learning using adaptive dynamic programming and reservoir computing," in *Re- inforcement Learning with Generalized Feedback: Beyond Numeric Rewards (ECML 2013)* (Prague).

Oubbati, M., and Palm, G. (2010). A neural framework for adaptive robot control. *Neural Comput. Appl.* 19, 103–114. doi: 10.1007/s00521-009-0262-2

Oubbati, M., Uhlemann, J., and Palm, G. (2012). "Adaptive learning in continuous environment using actor-critic design and echo-state networks," in *International Conference on Simulation of Adaptive Behavior* (Berlin; Heidelberg: Springer), 320–329.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evol. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Palm, G. (2012). *Novelty, Information and Surprise*. Heidelberg; New York, NY; Dordrecht; Berlin: Springer Science & Business Media.

Parisi, S., Pirotta, M., and Peters, J. (2017). Manifold-based multi-objective policy search with sample reuse. *Neurocomputing* 263, 3–14. doi: 10.1016/j.neucom.2016.11.094

Pirotta, M., Parisi, S., and Restelli, M. (2015). "Multi-objective reinforcement learning with continuous pareto frontier approximation," in *29th AAAI Conference on Artificial Intelligence, AAAI 2015 and the 27th Innovative Applications of Artificial Intelligence Conference, IAAI 2015* (Austin: AAAI Press), 2928–2934.

Pisula, W. (2009). *Curiosity and Information Seeking in Animal and Human Behavior*. Boca Raton, FL: Brown Walker Press.

Prokhorov, D. V., and Wunsch, D. C. (1997). Adaptive critic designs. *IEEE Trans. Neural Netw.* 8, 997–1007. doi: 10.1109/72.623201

Rescorla, R., and Wagner, A. (1972). "A theory of pavlovian conditioning: variations in the effectiveness of reinforcement and nonreinforcement," in *Classical Conditioning II: Current Research and Theory*, eds A. Black and W. Prokasy (New York, NY: Appleton Century Crofts), 64–99.

Riedmiller, M., Gabel, T., Hafner, R., and Lange, S. (2009). Reinforcement learning for robot soccer. *Auton. Robots* 27, 55–73. doi: 10.1007/s10514-009-9120-4

Roijers, D. M., Vamplew, P., Whiteson, S., and Dazeley, R. (2013). A survey of multi-objective sequential decision-making. *J. Artif. Int. Res.* 48, 67–113. doi: 10.1613/jair.3987

Schaal, S., and Schweighofer, N. (2005). Computational motor control in humans and robots. *Curr. Opin. Neurobiol.* 15, 675–682. doi: 10.1016/j.conb.2005.10.009

Scherer, S., Oubbati, M., Schwenker, F., and Palm, G. (2008). "Real-time emotion recognition from speech using echo state networks," in *IAPR Workshop on Artificial Neural Networks in Pattern Recognition* (Heidelberg; Berlin: Springer), 205–216.

Schmidhuber, J. (1991). "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, eds J. A. Meyer and S. W. Wilson (Cambridge, MA: MIT Press), 222–227.

Schmidhuber, J. (1997). *What's Interesting?* Technical Report 35-97, IDSIA.

Schmidhuber, J. (2003). "Exploring the predictable," in *Advances in Evolutionary Computing*, eds S. Ghosh and S. Tsuitsui (Cham: Springer), 579–612.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature* 529, 484–489. doi: 10.1038/nature16961

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). "Deterministic policy gradient algorithms," in *Proceedings of the 31 International Conference on Machine Learning* (Beijing).

Simon, H. A. (1955). A behavioral model of rational choice. *Q. J. Econ.* 69, 99–118. doi: 10.2307/1884852

Simon, H. A. (1991). Bounded rationality and organizational learning. *Organ. Sci.* 2, 125–134. doi: 10.1287/orsc.2.1.125

Sledge, I. J., and Príncipe, J. C. (2017). "Balancing exploration and exploitation in reinforcement learning using a value of information criterion," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on* (New Orleans, LA: IEEE), 2816–2820.

Sporns, O., and Pegors, T. K. (2003). "Information-theoretical aspects of embodied artificial intelligence," in *Embodied Artificial Intelligence, Volume 2865 of Lecture Notes in Computer Science* (Berlin; Heidelberg: Springer), 74–85.

Srinivasan, S., Lanctot, M., Zambaldi, V., Pérolat, J., Tuyls, K., Munos, R., et al. (2018). "Actor-critic policy optimization in partially observable multiagent environments," in *Advances in Neural Information Processing Systems* (Montreal, QC), 3426–3439.

Sutton, R. S., and Barto, A. G. (1981). An adaptive network that constructs and uses an internal model of its world. *Cogn. Brain Theory* 4, 217–246.

Sutton, R. S., and Barto, A. G. (1998). *Introduction to Reinforcement Learning, 1st Edn*. Cambridge, MA: MIT Press.

Tisdell, C. (1996). *Bounded Rationality and Economic Evolution*. Michigan: Edward Elgar Publishing.

Tishby, N., and Polani, D. (2011). "Information theory of decisions and actions," in *Perception-action Cycle*, eds V. Cutsuridis and A. Hussain, and J. G. Taylor (New York, NY: Springer), 601–636.

Todorov, E. (2004). Optimality principles in sensorimotor control. *Nat. Neurosci.* 7:907. doi: 10.1038/nn1309

Tokic, M., and Palm, G. (2012). "Adaptive exploration using stochastic neurons," in *International Conference on Artificial Neural Networks* (Berlin; Heidelberg: Springer), 42–49.

Tokic, M., Schwenker, F., and Palm, G. (2013). "Meta-learning of exploration and exploitation parameters with replacing eligibility traces," in *IAPR International Workshop on Partially Supervised Learning* (Berlin; Heidelberg: Springer), 68–79.

Uchibe, E., and Doya, K. (2008). Finding intrinsic rewards by embodied evolution and constrained reinforcement learning. *Neural Netw.* 21, 1447–1455. doi: 10.1016/j.neunet.2008.09.013

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., and Dekker, E. (2011). Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Mach. Learn.* 84, 51–80. doi: 10.1007/s10994-010-5232-5

Vamplew, P., Issabekov, R., Dazeley, R., Foale, C., Berry, A., Moore, T., et al. (2017). Steering approaches to Pareto-optimal multiobjective reinforcement learning. *Neurocomputing* 263, 26–38. doi: 10.1016/j.neucom.2016.08.152

Van Moffaert, K., and Nowé, A. (2014). Multi-objective reinforcement learning using sets of Pareto dominating policies. *J. Mach. Learn. Res.* 15, 3483–3512.

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., et al. (2001). Autonomous mental development by robots and animals. *Science* 291, 599–600. doi: 10.1126/science.291.5504.599

Wiering, M., and Schmidhuber, J. (1998). "Efficient model-based exploration," in *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, Vol. 6, (Cambridge, MA: MIT Press), 223–228.

Wiering, M., and van Otterlo, M. (2012). *Reinforcement Learning: State of the Art*. Heidelberg; New York, NY; Dordrecht: Springer. doi: 10.1007/978-3-642-27645-3

Wiering, M. A., and De Jong, E. D. (2007). "Computing optimal stationary policies for multi-objective markov decision processes," in *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on* (Honolulu, HI: IEEE), 158–165. doi: 10.1109/ADPRL.2007.368183

Wikenheiser, A. M., and Schoenbaum, G. (2016). Over the river, through the woods: cognitive maps in the hippocampus and orbitofrontal cortex. *Nat. Rev. Neurosci.* 17, 513–523. doi: 10.1038/nrn.2016.56

Wong, W. C., and Lee, J. H. (2010). A reinforcement learning-based scheme for direct adaptive optimal control of linear stochastic systems. *Opt. Cont. Appl. Methods* 31, 365–374. doi: 10.1002/oca.915

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH
published: 24 July 2019
doi: 10.3389/fnbot.2019.00056

Action Generation Adapted to Low-Level and High-Level Robot-Object Interaction States

Carlos Maestre[1]*, *Ghanim Mukhtar*[1], *Christophe Gonzales*[2] *and Stephane Doncieux*[1]

[1] UMR 7222, ISIR, Sorbonne Université and CNRS, Paris, France, [2] UMR 7606, LIP6, Sorbonne Université and CNRS, Paris, France

Our daily environments are complex, composed of objects with different features. These features can be categorized into low-level features, e.g., an object position or temperature, and high-level features resulting from a pre-processing of low-level features for decision purposes, e.g., a binary value saying if it is too hot to be grasped. Besides, our environments are dynamic, i.e., object states can change at any moment. Therefore, robots performing tasks in these environments must have the capacity to (i) identify the next action to execute based on the available low-level and high-level object states, and (ii) dynamically adapt their actions to state changes. We introduce a method named Interaction State-based Skill Learning (IS^{2}L), which builds skills to solve tasks in realistic environments. A skill is a Bayesian Network that infers actions composed of a sequence of movements of the robot's end-effector, which locally adapt to spatio-temporal perturbations using a dynamical system. In the current paper, an external agent performs one or more kinesthetic demonstrations of an action generating a dataset of high-level and low-level states of the robot and the environment objects. First, the method transforms each interaction to represent (i) the relationship between the robot and the object and (ii) the next robot end-effector movement to perform at consecutive instants of time. Then, the skill is built, i.e., the Bayesian network is learned. While generating an action this skill relies on the robot and object states to infer the next movement to execute. This movement selection gets inspired by a type of predictive models for action selection usually called affordances. The main contribution of this paper is combining the main features of dynamical systems and affordances in a unique method to build skills that solve tasks in realistic scenarios. More precisely, combining the low-level movement generation of the dynamical systems, to adapt to local perturbations, with the next movement selection simultaneously based on high-level and low-level states. This contribution was assessed in three experiments in realistic environments using both high-level and low-level states. The built skills solved the respective tasks relying on both types of states, and adapting to external perturbations.

Keywords: skill building, action generation, learning from demonstration, affordances, motor control, state, Bayesian inference, closed-loop

OPEN ACCESS

Edited by:
Gianluca Baldassarre, Italian National Research Council (CNR), Italy

Reviewed by:
Eiji Uchibe, Advanced Telecommunications Research Institute International (ATR), Japan
Emre Ugur, Boğaziçi University, Turkey

****Correspondence:***
Carlos Maestre
maestre@isir.upmc.fr

Received: *31 May 2018*
Accepted: *09 July 2019*
Published: *24 July 2019*

Citation:
Maestre C, Mukhtar G, Gonzales C and Doncieux S (2019) Action Generation Adapted to Low-Level and High-Level Robot-Object Interaction States. Front. Neurorobot. 13:56. doi: 10.3389/fnbot.2019.00056

1. INTRODUCTION

Autonomous robots are expected to help us in our daily tasks. In tasks involving objects, these robots must perform actions that result in a change of the object states, e.g., changing an object position or increasing its temperature. Therefore, in order to solve these tasks a robot must possess a repertoire of actions producing expected changes, called *effects*. The variability of environments to perform a task makes hard for a robot designer to foresee all the possible situations the robot can face and predefine an action for each case. For example, during the last DARPA Robotic Challenge (Atkeson et al., 2018) several robots failed to perform a trial due to the execution of built-in actions under incorrect circumstances. Therefore, it is plausible to think that a robot must develop its own behavioral capacities through interactions with the environment and learn when to use them.

Based on this principle, in our previous work (Maestre et al., 2017) we developed a method for a robot to build its own skills. A simulated Baxter robot endowed with our method executed an exploration of a static environment learning to push an object to specific positions of the environment. More precisely, the robot built a sensorimotor skill that generated actions producing different effects in the object states. We defined *state* as a feature that is relevant for a task, e.g., the object position; sensorimotor skill, or just *skill*, as the process transforming robot and object states into robot motor commands; and *action* as a sequence of movements of the robot's end-effector inferred in a closed-loop by a skill. The skill was implemented as a Bayesian Network (BN), a graphical representation of dependencies for probabilistic reasoning (Pearl, 1988). The exploration of the environment performed by the robot generated a dataset of robot-object interactions, henceforth *interactions*, that was used to learn both the network structure and the CPDs. The results showed that it was possible to build the skill through simple interactions with the object. However, both the exploration and the environment were constrained: the exploration was performed using predefined movements of a fix length in a two-dimensional environment that could be only modified by the robot actions. Besides, the generated push actions produced rough trajectories. Therefore, it was necessary to scale up the method for a robot to solve tasks in more realistic environments in which: (i) the robot environment is three-dimensional and dynamic, i.e., the object states can change at any moment independently of the robot actions; (ii) the task requires the use of complex actions, i.e., pick-and-place an object; (iii) action selection also implies abstract states, e.g., an object is hot or grasped; and (iv) actions are continuous and must adapt to changes in the environment.

In the current paper we introduce an extension of our previous method named Interaction State-based Skill Learning (IS^{2}L), which builds skills to reproduce effects on objects in realistic environments. The main features of the method are threefold: first, the method generates continuous actions in three-dimensional environments that locally adapt to spatio-temporal perturbations (Gribovskaya et al., 2011), similarly to Khansari-Zadeh and Billard (2014) and Paraschos et al. (2017). Spatial perturbations are those related to changes of the spatial values of a state. For example, changes of the initial position of the robot's end-effector w.r.t. the object position before the execution of an action, or changes of the object position during the execution. Temporal perturbations are those related to a change of the duration of an action, i.e., if the robot's end-effector gets stuck or delayed during the execution of the action. The adaptation to these spatio-temporal perturbations is performed through a data augmentation of the available interactions using a dynamical system called *diffeomorphism* (Perrin and Schlehuber-Caissier, 2016). This method proposes to apply a deformation to the motion space that generates a vector field converging to the expected trajectory to execute. And thus a robot action can recover from a perturbation executing the motion described by the vector field.

Second, a skill built with our method generates actions simultaneously relying on the low-level and high-level states of both the robot and the environment objects during the interactions. *High-level states* are those representing higher level concepts related to action selection, e.g., an object color or shape (Montesano et al., 2008). *Low-level states* are those related to the execution of an action, e.g., an object position (Calinon et al., 2010). An interaction is represented as a sequence of high-level and low-level states and the next robot movement to perform at different instants of time. Therefore, the action generation consists in, given an effect to reproduce and both types of states, choosing the next movement to perform among all the possible ones. Namely, the BN selects the movement with highest posterior probability. This movement selection gets inspired by a type of predictive models for action selection usually called *affordances* (Jamone et al., 2016; Zech et al., 2017). An affordance is initially defined as the actions an agent can afford to execute through direct perception of an object (Gibson, 1966, 1986). In robotics, it has been defined as the acquired relation of applying an action on an object to obtain an effect (Sahin et al., 2007).

Third, the method builds complex trajectories using *imitation learning* (Billard and Calinon, 2016), in which an external agent performs one or more kinesthetic demonstrations of an action generating a dataset of low-level states of the robot and the environment objects.

The main contribution of this paper is combining the main features of dynamical systems and affordances into a single method to build skills that solve tasks in realistic scenarios. More precisely, combining the low-level movement generation of the dynamical systems, to adapt to local perturbations, with the next action selection simultaneously based on high-level and low-level states.

Three experiments, of increasing complexity, were executed to assess the feasibility of the method to generate skills using both low-level and high-level states. In the first experiment, the robot pushed an object to a final position in different mazes, only using the object position (low-level states). In the second experiment, the robot grasped a croissant and released it in a pan using as information the object positions (low-level states) and if the croissant was grasped at an instant of time (high-level state). Finally, in the third experiment the robot had to heat the croissant to a certain temperature (high-level state) turning a stove on and off pressing a button (high-level state). These experiments were directly performed by a physical Baxter robot, showing that the method is able to generate skills to solve tasks in realistic environments.

The remainder of this paper is organized as follows. Section 2 describes the background and works related to our method. Section 3 describes IS^{2}L. Section 4 describes the experiments and obtained results. Section 5 provides some conclusions to this study and identifies some possible future research lines.

2. RELATED WORK

There is certainly a lack of works in the robotics literature combining action selection (using high-level states) with adaptive action execution (using low-level states). To the best of the authors' knowledge, Kroemer et al. (2012) is the only work combining these features. In this work, a pouring task experiment is executed, in which a robotic arm *grasps* a watering can and *pours* water into a glass. The main objective of this experiment is to use affordance knowledge to learn predictive models mapping subparts of objects to motion primitives based on direct perception. The main different between our work and the one presented by Kroemer et al. consists in that they focus on the low-level features of an object, i.e., its shape acquired using a point cloud, to select the next action to apply; whereas our work uses a simpler low-level representation of the object, i.e., its location represented as a position, combined with other high-level object features for the action selection. A positive aspect of their work is that the method directly uses a sensor information as input, providing richer object information, which can help to generate accurate interactions with the objects. However, in order to handle high-level features the method should be combined with another method working in parallel, adding a relevant complexity to the symtem.

The remainder of the section introduces works related to either selecting the next action to perform (based on predictive models) or building a skill to reproduce an action (based on imitation learning and motor control techniques) using either anthropomorphic robots or robotics arms.

2.1. Selecting the Next Action To Perform

In the works introduced in this section action selection either relies on affordance knowledge or are based on non-linear mappings from raw images to robot motor actions. Actions are usually considered as built-in knowledge, externally tailored by a designer, and they are executed in an open-loop. These works are only robust to spatial perturbations before the execution of an action, i.e., to the object position, not adapting the action to spatial and/or temporal perturbations during its execution. This offline spatial adaptation is usually externally hard-coded by the experiment designer. This low adaptation capability can result in the inability to scale up the executed experiments to realistic setups.

The works depicted in **Table 1** are categorized based on the classification available in Jamone et al. (2016). The relevant categories for the current work are *Pioneering works* representing those first studies where the initial insights to learn the relation between objects and actions were identified; *Representing the effects* is the category with more related works, including IS^{2}L, and extends the previous action-object relations to take into account the corresponding effect; *Multi-object interaction* represents affordances among several objects; and finally *Multi-step prediction* represents the use of affordances in high-level task planners to solve complex tasks.

The goal of the *pioneering works* (Krotkov, 1995; Fitzpatrick and Metta, 2003; Metta and Fitzpatrick, 2003; May et al., 2007) was identifying affordances observing the result obtained when applying an action on an object, e.g., *rollability*. Posterior works (Fitzpatrick et al., 2003; Stoytchev, 2005) made the first attempts to learn the relation between the action and the obtained result, trying to choose the best action to reproduce it. However, actions and effects were very simple. In contrast, the works *representing the effects* focus on learning an inverse model to reproduce a previously observed effect on an object. Dearden and Demiris (2005), Demiris and Dearden (2005), and Hart et al. (2005) are the first works to propose representing the forward and inverse models using Bayesian Networks (BN) in this context, used to play imitation games. Inspired by the previous works, Lopes et al. (2007), Montesano et al. (2008), Osório et al. (2010), and Chavez-Garcia et al. (2016) define an affordance as a BN representing the relation between *action, object* and *effect*. They provide built-in *grasp, tap,* and *touch* actions to also play imitation games. Similarly, other works also use built-in actions using different methods to learn affordances, as classification techniques (Ugur et al., 2009, 2011; Hermans et al., 2013), regression methods (Kopicki et al., 2011; Hermans et al., 2013; Hangl et al., 2016), neural networks (Ridge et al., 2010), dynamical BN (Mugan and Kuipers, 2012), among others. Multi-object interactions has gathered many research attention during the last years, mainly focused on the use of tools to reproduce effects on objects. Jain and Inamura (2011), Jain and Inamura (2013), Goncalves et al. (2014), and Goncalves et al. (2014) use a BN to model affordances to *push* and *pull* objects using tools with different features, whereas Dehban et al. (2016) and Dehban et al. (2017) use Denoising Autoencoders. Conversely to tool use, Szedmak et al. (2014) proposes to model the interactions of 83 objects with different features assisted by a human expert. In the previous works a repertoire of built-in actions was available for the affordance learning. Nevertheless, a couple of works by Ugur and his collaborators built this repertoire beforehand (Ugur et al., 2012, 2015a). In these works a built-in *generic swipe action* is available, which executes a trajectory of a robot's end-effector from a fixed initial position to the position of a close object. Therefore, for different object positions different trajectories are built. Nevertheless, the shape of these trajectories does not differ much among them, because of the use of the same heuristic to generate them. Other works in the same vein are Finn et al. (2016), Finn and Levine (2017), and Ebert et al. (2017), which use a deep learning technique called convolutional LSTM (Hochreiter and Schmidhuber, 1997) in order to predict the visual output of an action. These works build a repertoire of continuous *push* actions based on an exploration performing thousands of interactions of a robotic arm with a set of objects (see Wong, 2016 for a recent survey about applying deep learning techniques in robotics).

TABLE 1 | Comparison of actions used within the affordance literature, where *represents ambiguous information.

Type	Publication	Affordance learning method	AA	OffSP	OnSP	TP	PA	RA
Pioneering works	Krotkov, 1995	–	–	No	No	No	Yes	Poke
	May et al., 2007	–	–	No	No	No	No	Random
	Metta and Fitzpatrick, 2003, Fitzpatrick and Metta, 2003	–	–	Object position	No	No	Yes	Tap
	Fitzpatrick et al., 2003	PI	–	Object position	No	No	Yes	Tap
	Stoytchev, 2005	DT	–	Object position	No	No	No	Random
Representing the effects	Demiris and Dearden, 2005	BN	–	Object position	No	No	No	Random
	Hart et al., 2005	DRN	–	Object position	No	No	Yes	Grasp
	Lopes et al., 2007, Montesano et al., 2008, Osório et al., 2010	BN	–	Object position	No	No	Yes	Grasp, Tap, Touch
	Ugur et al., 2009, 2011	SVM	–	Object position	No	No	Yes	Push
	Ridge et al., 2010	NN	–	No	No	No	Yes	Push
	Kopicki et al., 2011	LWPR	–	Object position	No	No	No	Push
	Ugur et al., 2012, 2015a	SVM	–	Object position	No	No	No	Grasp, Hit, Drop, Tap
	Mugan and Kuipers, 2012	DBN	–	Object position	No	No	Yes	Grasp
	Hermans et al., 2013	SVR	–	Object position and orientation	No	No	Yes	Push
	Finn et al., 2016, Finn and Levine, 2017	LSTM	–	Object position and orientation	No	No	No	Push
	Ebert et al., 2017	LSTM	–	Object position and orientation	No	No	Yes, No	Lift, Push
	Hangl et al., 2016	MMR	–	Object position and orientation	No	No	Yes	Push, Flip
	Chavez-Garcia et al., 2017	GBN	–	Object position	No	No	Yes	Push, Grasp
	This work	BN	LH	Object position	Yes	Yes	No	Push, Grasp, Press
Multi-object interaction	Jain and Inamura, 2013	BN	–	Object position	No	No	Yes	Push, Pull
	Goncalves et al., 2014	BN	–	No*	No	No	Yes	Tap, Push, Pull
	Dehban et al., 2016, 2017	DA	–	No*	No	No	Yes	Push, Pull
Multi-step predictions	Omrčen et al., 2008, Krüger et al., 2011	NN	–	Object position and orientation	No	No	Yes	Poke, Push, Grasp
	Ugur et al., 2015b, Ugur and Piater, 2015	SVM	–	Object position	No	No	Yes	Pick, Place, Poke, Stack
	Antunes et al., 2016	BN	–	No*	No	No	Yes	Grasp, Release, Pull

Works are categorized based on the classification available in Jamone et al. (2016) (see column Type). They are described based on the following features: Affordance learning method, AA, Action adaptation; OffSP, Offline Spatial Perturbation; OnSP, Online Spatial Perturbation; TP, Temporal Perturbation; BA, Built-in actions; RA, Repertoire of actions. The affordance learning methods are PI, Probabilistic Inference; DT, Decision Tree; BN, Bayesian Network; DRN, Relational Dependency Network; SVM, Support Vector Machine; NN, Neural Network; LWPR, Locally Weighted Projection Regression; DBN, Dynamic Bayesian Network; SVR, Support Vector Regression; LSTM, Long Short-term Memory; MMR, Maximum Margin Regression; GBN, Gaussian Bayesian Network; DA, Denoisy autoencoder.

2.2. Reproducing an Action

A robot can *learn from demonstration* all the actions required to reach a task goal. This section presents some of the most relevant works building skills, also called motion primitives, reproducing an action from one or more demonstrations. In **Table 2** there is a comparison of these works. The variables selected for the comparison represent the capability of a skill to adapt to low-level (L) and high-level states (H), together with the main features studied within the motor control literature: mechanisms to be robust to spatio-temporal low-level perturbations, the stability of a motion primitive, the number of examples needed for the learning, and the combination of different primitives to reproduce an unseen action.

Paraschos categorizes motion primitives as *trajectory-based representations*, which typically use time as the driving force of the movement requiring simple controllers, and *state-based representations*, which do not require the knowledge of a time step but often need to use more complex, non-linear policies. Paraschos et al. (2017, p. 2). On the one hand, trajectory-based primitives are based on dynamical systems representing motion as time-independent functions. The principal disadvantage of dynamical systems is that they do not ensure the stability of the system. In order to address this issue, an external stabilizer

TABLE 2 | Comparison of methods generating adaptive skills.

Type	Publication	MP learning method	AA	Spatial perturbation	Temporal perturbation	TD	St	NE	C
Trajectory-based	Ijspeert et al., 2002, Ijspeert et al., 2013	DMP	L	Final position	No	Yes	Yes	1	No
	Pastor et al., 2009, Kober et al., 2010	DMP	L	Final position and velocity	No	Yes	Yes	1	No
	Muelling et al., 2013	MoMP	L	Final position and velocity	No	Yes	Yes	1	Yes
	Paraschos et al., 2013, Paraschos et al., 2017	ProMP	L	All positions and velocities	Yes	No	Yes	M	Yes
	Perrin and Schlehuber-Caissier, 2016	Diffeomorphism	L	Final position	Yes	No	Yes	1	No
State-based	Calinon et al., 2007	GMR-DS	L	No	Yes	No	No	M	–
	Calinon et al., 2010, Calinon et al., 2011	HMM + GMR	L	Final position	Yes	No	No	M	–
	Khansari-Zadeh and Billard, 2011, Khansari-Zadeh and Billard, 2014, Kim et al., 2014	SEDS	L	Final position	Yes	No	Yes	M	–
	Calinon, 2016	TP-GMM	L	All positions and orientations	Yes	No	Yes	M	–
	This work	IS^{2}L	HL	All positions	Yes	No	No	M	Yes

Works are categorized based on the classification available in Paraschos et al. (2017) (see column Type). They are described based on the following features: MP, Motion primitive; AA, Action adaptation; SP, Spatial perturbation; TP, Temporal perturbation; TD, Time-dependency; St, Stable; NE, Number of examples; C, Combination of MPs.

based on time to generate stable motion is used (e.g., DMPs, Ijspeert et al., 2002, 2013; Pastor et al., 2009; Muelling et al., 2013). Therefore, actions are always executed following a specific time frame. A more recent approach called ProMP (Paraschos et al., 2013, 2017) avoids the previous constraint by generating time-independent stable primitives.

On the other hand, state-based motion primitives are time-independent by definition, in which the states use continuous values and are represented by Gaussian functions. For a specific position of the robot's end-effector, weights are computed using Hidden Markov Models (HMM) to identify the next state based on the current state. Once the state is available, the motion is computed using Gaussian Mixture Regression (GMR). The initial works (Calinon et al., 2007, 2010, 2011) do not generate stable actions, but it has been solved in posterior studies by a method called Stable Estimator of Dynamical Systems (SEDS) (Khansari-Zadeh and Billard, 2011, 2014; Kim et al., 2014), which ensures stability through a computation of Lyapunov candidates (Slotine and Li, 1991). However, SEDS can only handle spatial perturbations at the final position of the demonstrated trajectories. This feature is improved in Calinon (2016) handling spatial perturbations at any position of the trajectory, through the generation of a set of waypoints around the trajectory with different reference frames.

As aforementioned, works in the literature focus on either selecting the next action to perform a task based on high-level states using predefined or constrained actions; or in the reproduction with local adaptation of the trajectories of a complex action using low-level object states. Therefore, the skills built by IS^{2}L are unique to infer actions with local adaptation simultaneously based on both types of states.

3. INTERACTION STATE-BASED SKILL LEARNING (IS^{2}L)

This section explains the method Interaction State-based Skill Learning (IS^{2}L). Given several examples of a robot performing an action, i.e., producing a specific effect, on an object, the method creates a skill that generates actions reproducing the effect on the object. These actions can cope with local changes in the position of the object. At the left side of **Figure 1** a flowchart of the steps of the method is available. The method is based on the interactions between the robot's end-effector performing the action and the object. An interaction represents a sequence of the robot and object states during a period of time. More precisely, at each instant of time the high-level states of the robot and the object, and the low-level state representing the relative position of the object with respect to the robot, called *relation state*, are represented.

Each interaction is composed as a sequence of (i) high-level states of the robot and the object, and (ii) the low-level state representing the relative position of the object with respect to the robot, called *relation state*, at different instants of time. Robot actions and object effects represent the difference of these states between two consecutive instants of time. The main advantage of this approach is that the method does not build skills reproducing an interaction in a specific scenario. These skills use the most relevant information during the interactions, i.e., high-level states and relation states, to infer actions under similar robot-object interactions with local adaptation to perturbations.

A skill is a BN that, given an effect to reproduce and a relation state, infers the next robot movement to perform (see the next sections for further details). In order to simultaneously handle high-level and low-level states, the BN uses discrete values,

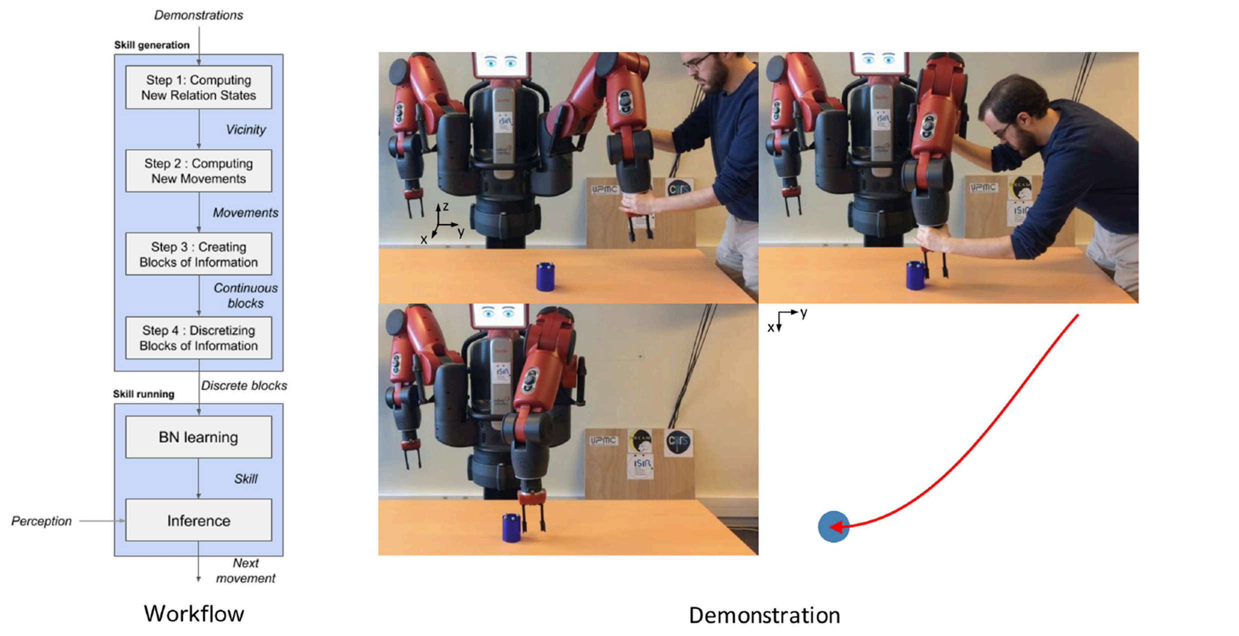


FIGURE 1 | On the left, a flowchart of the steps of the method. On the right, an initial kinesthetic demonstration of a trajectory *pushing* an object. At the top-left corner, the setup of the experiment. At the top-right corner, the demonstration performed by a co-author of this paper. At the bottom-left, the object was pushed certain distance and orientation. At the bottom-right, top-view graphical representation of the action. The red arrow represents the demonstrated trajectory, and the blue circle represents the final position of the object. Although the reference frame of the setup is located in the base of the robot, in order to facilitate the visual comprehension of the setup the reference frames are depicted in different places.

although the inferred robot action is continuous. In the current paper, the set of actions a robot can perform is composed of *push*, *grasp*, *release*, *set* and *press*.

3.1. Initial Available Information

Some available information is needed to execute the method. First, interactions must represent the relevant states to perform different actions on objects. IS^{2}L relies on a previous developmental stage identifying these states that E. J. Gibson calls *differentiation* (Gibson, 2000, 2003), which is out of the scope of our work (a recent and relevant approach is available in Jonschkowski and Brock, 2015; Jonschkowski et al., 2017).

Second, some a priori information is needed to build a skill. A BN is a graphical representation of dependencies for probabilistic reasoning, in which the nodes represent *random variables* and the lack of arcs represent *conditional independence relationships* between the variables (Pearl, 1988). More precisely, a BN is a directed acyclic graph, i.e., a collection of nodes or vertices joined by directed edges without directed cycles. Besides the structure, which provides qualitative information about the probabilistic dependencies between the variables, a BN also encodes quantitative information about the strength of these dependencies through Conditional Probabilistic Distributions (CPDs). In the current work, the structure represents the knowledge that an interaction is based on the relative position of the end-effector and an object, and the actual values of the interaction are stored as CPDs. In our previous work (Maestre et al., 2017) a simulated Baxter robot executed an exploration of a static environment identifying the BN structure and CPDs to *push* an object in different directions. The results demonstrated that the BN structure is task- and environment-agnostic. Therefore, in the current paper the structure is provided. And thus building a skill consists in learning the correct CPDs to reproduce an effect. Second, in our previous work we also identified a generic discretization configuration to discretise the relation states (explained at the end of section 3.2).

Finally, a dataset of interaction demonstrations, $\mathbb{D}$, must be available to build skills (at the right side of **Figure 1**, a demonstration of a *push* action). For the aforementioned list of possible actions, the low-level states represent at each instant of time the end-effector position, x_t, and the object position, y_t; whereas the high-level states represent at each instant of time the discrete gripper openness (open/closed), G_t, and object high-level states, H_t, representing different object features[1]. Therefore, an interaction, Υ_{xygh}, is represented as:

$$\begin{aligned} x_t &= \text{end effector position} \\ G_t &= \text{gripper state} \\ y_t &= \text{object position} \\ H_t &= \text{high-level object states} \\ \Upsilon_{xygh} &= \{(x_0, y_0, G_0, h_0), ..., (x_T, y_T, G_T, h_T)\} \end{aligned} \tag{1}$$

[1]In the mathematical formalization uppercase variables represent states with discrete values, whereas lowercase variables represent states with continuous values.

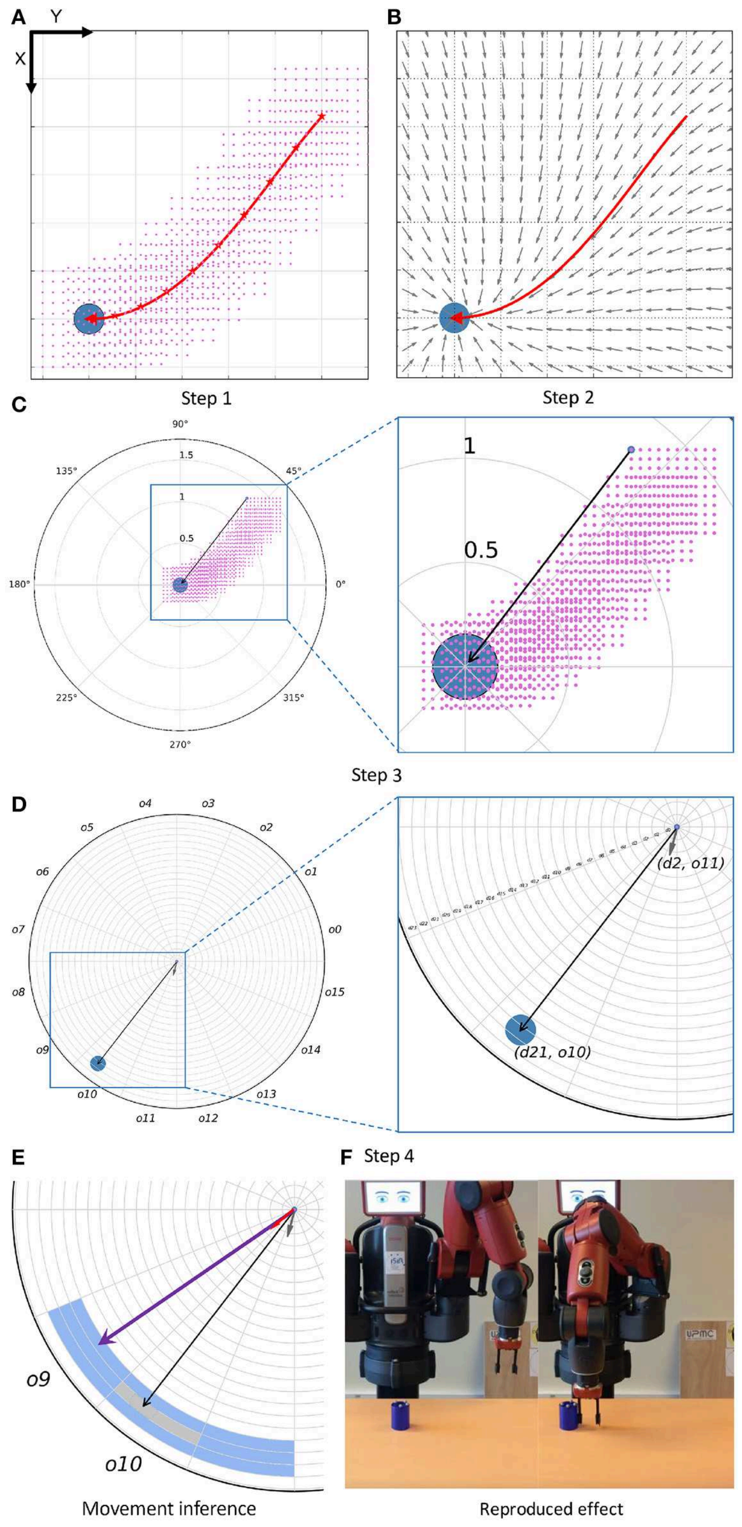


FIGURE 2 | (A) Example of the computation of a vicinity. The trajectory is performed in the Cartesian X-Y plane, and the figure represents the top view of the setup. The trajectory is represented with a red arrow, and the object with a blue circle. The red stars represent the waypoints selected to compute the vicinity of the trajectory. For each of them, a set of end-effector positions is generated, represented by the pink points. **(B)** Example of the computation of movements from new positions of the end-effector. Each gray arrow represents the next movement, from a position, to be executed by the end-effector in order to reproduce the demonstrated effect. **(C)** Example of the computation of a continuous relation state (black arrow) a position of the vicinity of the trajectory (selected with a blue circle). **(D)** Example of the

(Continued)

FIGURE 2 | discretization of a block of information, i.e., discretizaton of both the continuous relation state and movement (gray arrow) into a *distance* and an *orientation*. In this case, the discrete relation state has values *d21* and *o10*, whereas the discrete movement has values *d2* and *o11*. **(E)** Example of the inference of a movement by the skill. A neighbor of the previous block of information is depicted, being its relation state represented as a purple vector (*d21*, *o9*) and its movement as a red vector (*d2*, *o9*). **(F)** Initial and final instants (from left to right) of the reproduction of the demonstrated effect.

An effect is defined as an expected variation of the object states, $\widehat{\Lambda f}$, in between two instants of time, *t* and *t-1*, and it is associated to a label, *e*. The effect can be reproduced multiple times, and thus it is not related to any specific instant of time. In the current work the expected variation can be related to either a variation of the object position or a variation of the high-level object states:

$$e \equiv \widehat{\Lambda f} = y_t - y_{t-1} \vee H_t - H_{t-1}$$

where the subscript *t* represents an instant of time.

Therefore, a dataset of interactions is represented as:

$$\mathbb{D} = (e, \{\Upsilon^k_{xygh}\}) \quad (2)$$

where *k* represents one of the *K* interactions available.

3.2. Skill Generation

Once the dataset of interactions is available the method to build the skill starts, which is composed of two processes:

- *Dataset augmentation and transformation*: first, the dataset of interactions, $\mathbb{D}$, is extended and transformed into a repertoire, *R*, of discrete *blocks* of information (section 3.2.1).
- *Skill building*: second, the skill is built based on the dataset of blocks, i.e., the CPDs of the BN are learned (section 3.2.2).

3.2.1. Dataset Augmentation and Transformation

The dataset of interactions, $\mathbb{D}$, represents one or more interactions producing an effect on an object. This sections explains the initial interaction augmentation and their posterior transformation into a sequence of blocks of information. A block of information, *B*, represents the relation of some high-level states to some low-level states at an instant of time to reproduce an effect on an object. More concretely, each block is a triple composed of (i) the relation state at an instant of time, δ, (ii) the high-level states of the robot and the object at that instant, *H*, and (iii) the next movement of the end-effector to execute reproducing an effect in an object, $(\Lambda x_t, \Lambda g_t)$:

$$B = (\delta, H, (\Lambda x_t, \Lambda G_t))$$

$$R = \{B\}$$

where Λ represents a difference of value of a variable between two instants of time, *t* and *t-1*.

Once *R* is available the CPDs can be learned, reproducing the same actions that were demonstrated and captured in $\mathbb{D}$. However, with the current dataset if the robot faces an unobserved relation state, for example due to noise in the actuators of the robot or external forces, the BN would not be able to infer any movement. Namely, the skill is not yet robust to spatio-temporal perturbations.

It would be highly expensive to record interactions of the robot reproducing an effect from very similar relation states. Therefore, the method generates an augmentation of the blocks in $\mathbb{D}$ adding*different but close* relation states. The approach is inspired from Calinon et al. (2010), where a set of Gaussians is computed along a demonstrated trajectory describing end-effector movements converging to the trajectory. IS^{2}L computes a sampling of positions of the end-effector around the trajectory of the demonstrated interaction generating the new relations states, called *vicinity* (Step 1). Then, for each new relation state of the vicinity an end-effector movement is computed using a dynamical system (Step 2), and a new discrete block of information is stored into *R* (Steps 3 and 4).

Step 1: Computing New Relation States Using a Vicinity. A vicinity is computed for the trajectory of each demonstrated interaction. First, the trajectory is reduced to a set of equidistant waypoints (represented as red stars in the Step 1 of the **Figure 2**). The number of waypoints is computed based on the length of the trajectory. The higher the number of waypoints, the more precise the representation of the demonstration. However, a very high number of waypoints can affect the speed in which the BN infers a movement, because of the size of the CPDs. For each waypoint a vicinity is created, i.e., a sampling of unobserved end-effector positions. A vicinity is represented as a cubic grid centered in the waypoint with side size Q, and composed of *P* x *P* x *P* equidistant positions, *P* and *Q* being preset values. For each position of a vicinity, i.e., for each new end-effector position, a relation state is computed.

Step 2: Computing End-effector Movements for the New Relation States. Similarly, for each position of the vicinity the corresponding movement of the end-effector is computed using a *vector field*. This field generates a vector, i.e., a movement, for any position of the end-effector. End-effector positions close to the trajectory generate similar movements to the trajectory, whereas far end-effector positions generate movements less similar to those of the trajectory, mainly oriented to its end. Therefore, only those positions in the vicinity of the trajectory are relevant to reproduce an effect. An example of a vector field is depicted in the Step 2 of **Figure 2**.

In the current work, vector fields are generated using a dynamical system called *diffeomorphism* (Perrin and Schlehuber-Caissier, 2016). This method proposes to apply a deformation to the motion space in order to fit a simple trajectory to a demonstrated interaction trajectory. More precisely, the approach aims to minimize a defined distance between both trajectories using a diffeomorphic matching algorithm. This dynamical system has a parameter to compute the tendency to reproduce the demonstrated trajectory. As the possible actions generated by our method share similar features, i.e., they are based on interactions of a gripper and an object, the parameter value is empirically preset.

Step 3: Creating The Blocks of Information. Once both the new relation states and the robot movements are available, the new blocks of information are created. To that end, the high-level states related to each waypoint of the trajectory, W, are correlated to the relation states and movements created in the corresponding vicinity, V. Therefore, for each position of V a new block is created, composed of (i) the robot and object high-level states at waypoint W, (ii) the relation state computed from that position and (iii) the robot movement computed from the same position.

Step 4: Discretizing the Blocks of Information. The BN needs discrete information to infer a discrete movement. Therefore, each block of information is discretized before being stored into R. As the high-level information is already discrete, only the relation states and the movements are discretized. Both are vectors defined in a three-dimensional Cartesian space, composed of a *distance*, an *orientation* and an *inclination*. However, vector discretization in the Cartesian coordinates is complex, due to the range of each axis is $[-\infty, \infty]$. For this reason these vectors are transformed to *spherical coordinates* before being discretized. A vector in spherical coordinates is composed of a *distance*, with range $[0, \infty]$, an *orientation*, with range, $[-\pi, \pi]$ and an *inclination*, with range $[0, \pi]$. In the current work, the range of the *distance* is limited to the maximal reach distance of the robot's end-effector, i.e., 0.5. The values for the *orientation* and *inclination* are predefined based on experience, i.e., their ranges are divided into a preset number of *bins* of the same size. However, the distance size is task-agnostic because it determines the accuracy of the movements. For the available set of actions the distance of each movement is computed w.r.t. to the distance between two positions in the vicinity:

$$\textit{minimal distance bin size} = \frac{Q}{P-1} \tag{3}$$

3.2.2. Building the Skill

Once the discrete repertoire of blocks, R, is available, the skill is built. As aforementioned, a skill, ϕ, is a BN that infers discrete movements, ΛX, to reproduce a discrete effect, E, on an object. Each movement is generated w.r.t to both the discrete relation state, δ, and the discrete high-level robot and object states, H, at certain instant of time. In parallel to the inference of the movement, the method also infers the next open/close action of the end-effector based on the robot high-level state, if the skill is related to the *grasp* action.

Movement and gripper actions are independently inferred:

$$\begin{aligned}(\Lambda X_t, \Lambda G_t) &= \phi(E, \delta, H)\\ \Lambda X_t &= \underset{\Lambda X_t}{\arg\max}\, P(\Lambda X_t \mid E, \delta, H)\\ \Lambda G_t &= \underset{\Lambda G_t}{\arg\max}\, P(\Lambda G_t \mid E, \delta, H)\end{aligned} \tag{4}$$

A discrete movement is described using three discrete values, i.e., the distance, the orientation and the inclination:

$$\Lambda X_t = (\Lambda_{dist} X_t, \Lambda_{orien} X_t, \Lambda_{inclin} X_t)$$

Although it is possible that there is a weak dependency among these values, in order to speed up the computation of a movement we consider that these values are independent. And thus the inference of a movement consists in the individual inference of each one of them:

$$\begin{aligned}\Lambda X_t = (&\underset{\Lambda_{dist} X_t}{\arg\max}\, P(\Lambda_{dist} X_t \mid E, \delta, H),\\ &\underset{\Lambda_{orien} X_t}{\arg\max}\, P(\Lambda_{orien} X_t \mid E, \delta, H),\\ &\underset{\Lambda_{inclin} X_t}{\arg\max}\, P(\Lambda_{inclin} X_t \mid E, \delta, H))\end{aligned} \tag{5}$$

A relevant feature of our method is that skills directly combine information from different demonstrations. More precisely, the discrete repertoire of blocks, R, can contain information of one or more interactions, i.e., they have been computed based on trajectories of different demonstrations. The blocks of information generated from the different trajectories are stored into the same repertoire of blocks. And thus the related skill can infer movements combining information from different demonstrations (see **Figure 3**). It may happen that for the same relation state more than one movement have been stored in the same repertoire. These cases are directly handled by the probability distributions of the BN, calculating different probabilities for each movement.

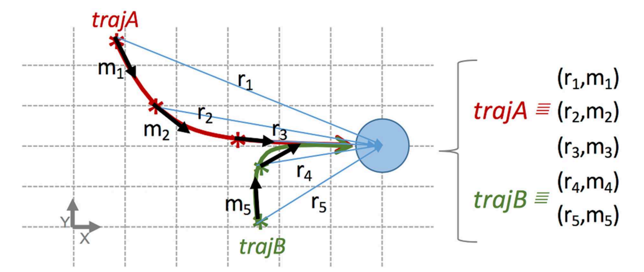


FIGURE 3 | Example transforming two interactions into blocks of information stored in the same repertoire, used to build a skill reproducing the same effect from two different initial relation states (in order to facilitate the comprehension, only low-level states are used). Each block is represented as a tuple (relation state, movement), e.g., r_1, m_1. The figure represents a top-view in a table-top scenario similar to the demonstration in **Figure 1**. The blue circle represents an object. The robot pushes the object to the right from two initial positions of its end-effector. More precisely, the trajectory of two interactions, A (red) and B (green), reproduce the same effect from two different initial relation states. Each trajectory is split up into few blocks stored in the same dataset. In this case, the trajectory A is split up in blocks 1, 2, and 3; and the trajectory B in blocks 4 and 5. All blocks are mutually independent from temporal and spatial point of views. Namely, with this dataset our method is able to build a skill inferring the next movement to push the object to the right from 5 different relation states.

3.3. Reproducing an Effect on an Object

When a skill is available, the inference and execution of each movement is performed within a *perception-action cycle* (Kugler and Turvey, 1987; Warren, 1988). In a continuous loop, the perceptual information acquired by the robot's sensors is transformed into high-level and relation states and provided as input to the BN, which infers a movement. Then, the movement is executed by the robot using its inverse kinematic model. This execution generates a displacement of the position of the robot's end-effector, which can modify the robot's environment. If the effect has not been reproduced, or a maximum number of movements executed, a new iteration of the cycle is executed.

It may happen, depending on the vicinity parameters, that while reproducing an effect the end-effector moves to a position whose relation state is not stored into the repertoire of blocks, and thus the skill would not infer any movement and the effect would not be reproduced. Instead of identifying a task-dependent discretization configuration to cover all the possible relation states, movements are computed as the mean value of a set of relation states, Ξ. This set consists of the *nearest neighbors* relation states of the current relation state, including itself. For each dimension of the vector state (the distance d, the orientation o and the inclination c) the neighbors are the previous and the next relation bins based on the discretization configuration:

$$\begin{aligned} \textit{mean relation state} \; &= \; [(\sum_{a=i-N}^{i+N} d_a)/(N*2) \\ &+1, (\sum_{u=i-N}^{i+N} o_u)/(N*2)+1, (\sum_{q=i-N}^{i+N} c_q)/(N*2)+1] \end{aligned}$$

where *f, g, h* represents the number of the current bin, and N represents the number of neighbors at each side of the current bin. An example of this computation only using a distance and an orientation is available in the movement inference of **Figure 2**. In this example the current relation state is *d21* and *o10*. For one neighbor, *N=1*, the ranges of nearest neighbors would be [*d20, d21, d22*] for the *distance*, and [*o9, o10, o11*] for the *orientation*. And thus the computed mean relation state [(d20 + d21 + d22) / 3, (o9 + o10 + o11) / 3] would be used, together with the high-level states to infer the next movement.

Once a discrete movement has been inferred it is transformed into a continuous movement to be executed by the robot end-effector. This process simply selects the mid value of the range corresponding to each dimension composing the movement. For example, for the movement (*d2, o11*) the function computes the mid value for the bins *d2* and *o11*.

4. EXPERIMENTAL FRAMEWORK

Three experiments, of increasing complexity, were executed to assess the feasibility of the method to generate skills using both low-level and high-level states (see **Table 3**). In the first experiment, the robot pushed an object to a final position in different mazes, only using the object position (low-level states). In the second experiment, the robot grasped a croissant and released it in a pan using as information the object positions (low-level states) and if the croissant was grasped at an instant of time (high-level state). Finally, in the third experiment the robot had to heat the croissant to a certain temperature (high-level state) turning a stove on and off pressing a button (high-level state).

Figure 4 shows the set of objects used for the experiments. The positions of the objects were acquired using an OptiTrack motion capture system[2], composed of 4 cameras located at the ceiling, over the robotic setup. This system generated a virtual representation of each object, providing its center position, using markers located on it. The reference frame of the experimental setup was located at the base of the robot, and thus the object positions were relative to itself.

The validation of the method was performed on a physical Baxter robot. Each gripper of the robot had a different configuration: on the left gripper, the fingers of the gripper were in the farthest position, in order to grasp big objects; on the right gripper, the fingers were in a intermediate position, in order to grasp smaller objects. Both grippers had finger adapters in order to facilitate the corresponding targeted actions. The execution of the robot relied on ROS Indigo Igloo and our kinematic library[3]. Videos of the experiments are available online[4].

4.1. A Priori Knowledge

One or more demonstrations were performed for each one of the skills used in the experiments, i.e., *push*, *set*, *grasp*, *release* and *press*. As aforementioned in the Step 1 of section 3.2.1, the accuracy of an action is based on the number of positions, P, and the size of the vicinity, Q, used to transform the demonstrations for the CPD learning. Based on these values two BNs with different levels of accuracy were learned using the available demonstrations (see **Figure 5C**): (i) a coarse-grained BN inferring bigger movements (around 6 cm) with P equal to 8 positions and Q equal to 40 cm, approaching the end-effector to the object; (ii) a fine-grained BN inferring small and more accurate movements (around 2.5 cm) with P equal to 7 positions and Q equal to 20 cm. These values were chosen based on experience. The fine-grained generator was used if the end-effector was close to an object (arbitrarily preset to 10 cm), whereas the coarse-grained generator was used in any other case. The gripper state was either *open* if its openness value was in its top half range, i.e., 50 or more over 100, or *closed* otherwise. **Figures 5A,B** show the structure of the learned BNs for the *push* and *grasp* skills. Some nodes represent the robot and object state before the execution of the movement: the nodes *distance*, *orientation*, *inclination* represent the relation state; and the node *grasped* represents if the object is grasped. The other nodes represent the movement to perform: the nodes *move_dist*, *move_orien* and *move_inclin* represent the end-effector movement; whereas *next_openness* represents the openness of the end-effector grippper.

For the discretization configuration, the distance had a range of [0, M], where M represents the longest distance of a movement

[2]http://optitrack.com/
[3]https://github.com/cmaestre/baxter_kinematics
[4]https://www.youtube.com/playlist?list=PL2drYAFCMtzf4AC_ZRZjk8lNv2Zs9fh5Z

TABLE 3 | Skills, objects and object states used in the experiments (LL and HL stand for low-level and high-level states, respectively).

ID	Experiment	Skills	Objects	Object LL	Object HL	Robot LL	Robot HL
1	Solving a Maze	Push Set	Cylinder Cake	X		X	
2	Grasping a Croissant with Spatio-temporal Perturbations	Grasp Release	Croissant Pan Dish Button	X		X	X
3	Heating a Croissant	Grasp Release Press	Croissant Pan Dish Button	X	X	X	X

of the robot, in this case 50 cm. This range was discretized in bins of the same size, which size is computed as in Equation 3. Finally, both the orientation and the inclination were split up in 16 bins of the same size.

4.2. Experiments

4.2.1. Experiment 1: Solving a Maze

4.2.1.1. Experimental Setup

A table of 180 × 80 × 75 cm of width, length, and height, respectively, was located in front of the Baxer robot (see **Figure 6**). The setup of this experiment consists of two mazes of different configurations. The objects to *push*, i.e., the cylinder for the first maze and the cake for the second maze, have different sizes, shapes and weights.

4.2.1.2. Description

The task consisted in *pushing* an object through a maze to a final position. In these experiments the experiment designer chose the next action to execute and the distance to move the object, i.e., the effect to reproduce. Therefore, the goal of this experiment was to validate that the generated skills were able to reproduce an effect only relying on the object and gripper positions, i.e., low-level states. Besides, the experiments also validated the reproduction of different effects for the same skill, e.g., *pushing* an object to the right different distances.

In order to reproduce the sequence of actions different skills were demonstrated to the robot. First, a set of demonstrations were executed to *push* an object to the *left*, to the *right*, *close* to the robot, and *far* from the robot. Before executing each *push* action it is necessary to set the robot's end-effector on one side of the object, e.g., to *push* it to the right the end-effector must be located at the left of the object. Therefore, a set of demonstrations were executed to move the end-effector from the object to one of its sides (for example the C-D action on the top of **Figure 8**).

4.2.2. Experiment 2: Grasping a Croissant With Spatio-Temporal Perturbations

4.2.2.1. Experimental Setup

The scenario comprised a toy-like kitchen and other objects on it (see **Figure 7**). The kitchen, located in front of the robot, was composed of four stoves, a dish, a pan, a croissant and a switch button. The switch button turned on and off the stoves. This scenario was also used in the Experiment 3.

4.2.2.2. Description

Two tests were carried out in order to validate reproducing effects using simultaneously both low-level features, i.e., the robot and object position, and high-level states, i.e., the *openness* state of the end-gripper. Also, the tests had to validate the robustness of the skills with respect to spatio-temporal perturbations of the low-level states. In the first test, the robot had to *grasp* a croissant and *release* it inside a pan. The position of the croissant changed during the *grasp* action whereas the pan position changed during the *release* action (see **Figure 9a**). The second test consisted in *grasping* the croissant. During the execution of the *grasp* action either the position of the end-effector was externally modified or the croissant position changed (see **Figure 9b**). In both tests the designer produced the perturbations.

4.2.3. Experiment 3: Heating a Croissant

4.2.3.1. Description

The objective of this experiment was to show that skills built by IS^{2}L can be used to perform a multi-step task in a realistic scenario, simultaneously relying on high-level and low-level states of both the robot and the objects. The task consisted in heating a croissant in a pan until reaching a specific temperature. The high-level states of the objects were:

- Stove number 4 : *on* (red) or *off* (black).
- Croissant: *cold* (yellow), *hot* (salmon) or grasped (green).
- Button: *pressed* or *not pressed*.

The different state colors were visually represented during the experiment in a screen next to the robot (see **Figure 10**). Initially, the stove was *off*, the button was *not pressed*, the *croissant* was *cold* and located in the dish, over the stove 1 (which was always off). If the croissant was in the pan, the pan was over the stove 4, and the stove was *on*, the temperature of the croissant changed from *cold* to *mid temperature* after few seconds; and from *mid temperature* to *high temperature* again after few seconds.

The available repertoire of actions were *pressing* the button, *grasping* the croissant, and *releasing* the croissant from the dish

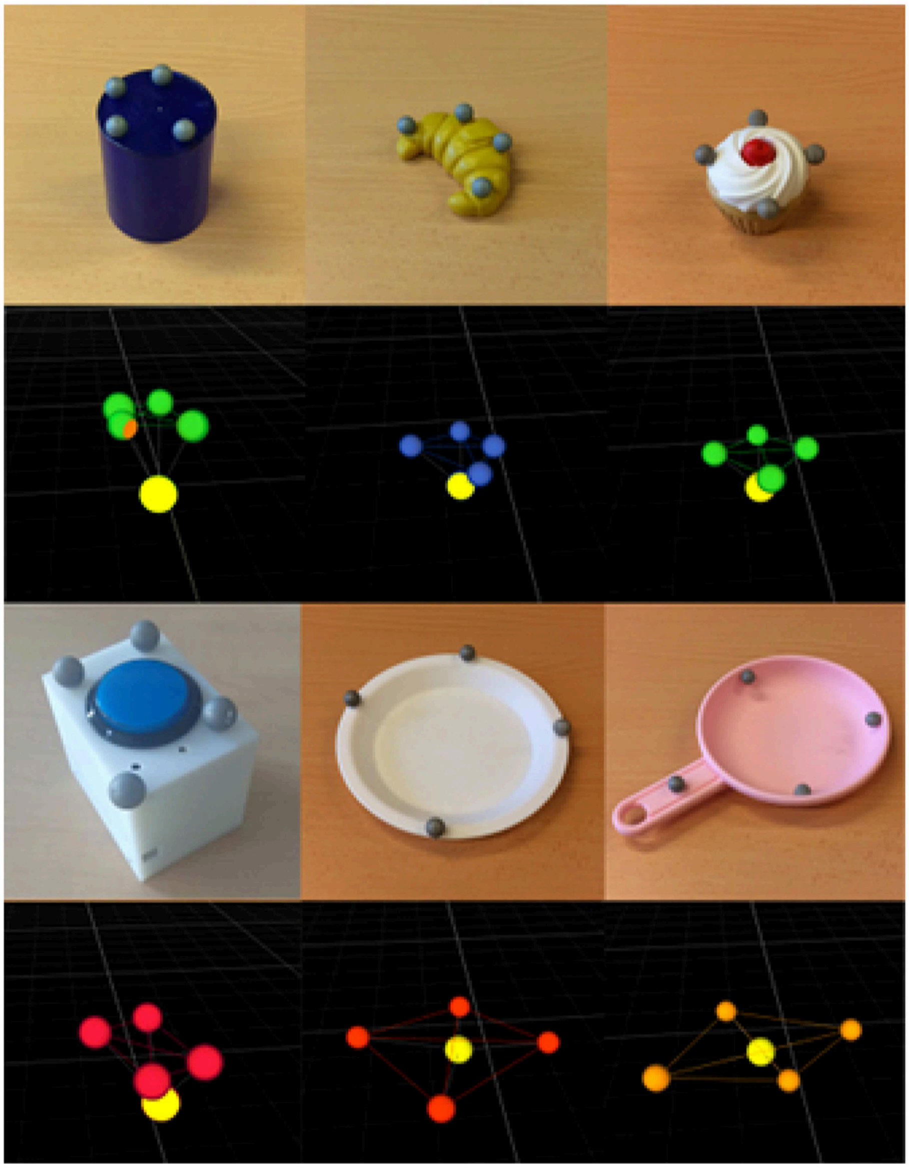


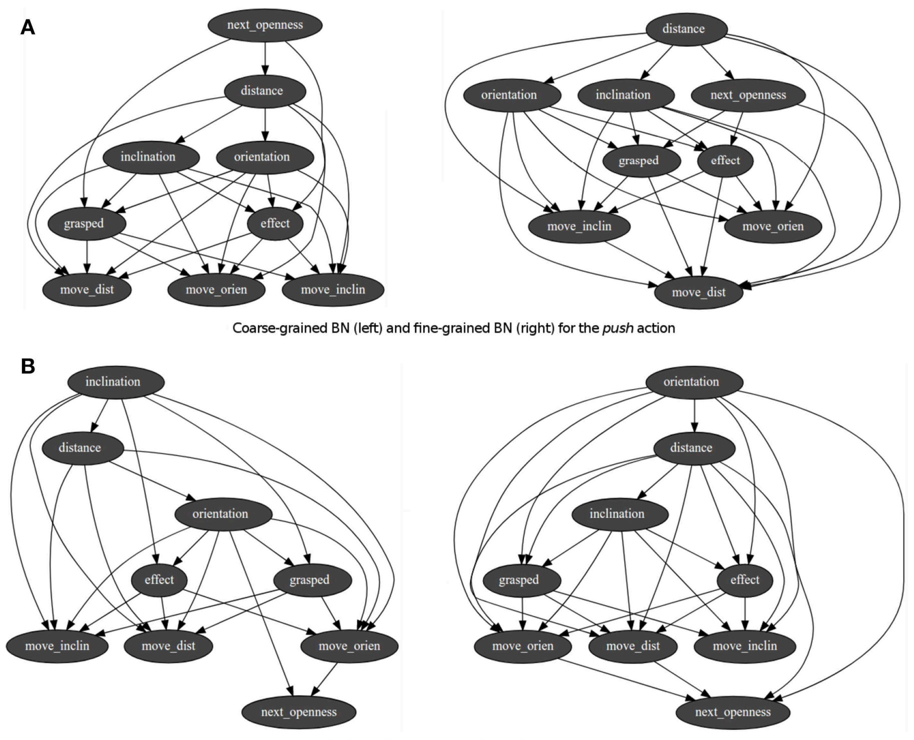
FIGURE 4 | Set of objects used in the experiments, i.e., a cylinder, a croissant, a cake, a button, a dish and a pan, respectively. For each object a photo (at the top) and its representation captured by the OptiTrack system (at the bottom) are provided. The light yellow marker in each structure represents the center position of the object acquired by the robot.

to the pan, and vice versa. Before the *grasp* and *press* actions the end-effector was randomly located over the setup, in a range of 20–40 cm of height, in order to show that actions can be inferred from different initial positions of the end-effector. The sequence of actions to reach the task goal was:

1. *Push* the button to turn the stoves on.
2. *Grasp* the croissant.
3. *Release* it into the pan.
4. When the croissant is hot *grasp* it again.
5. *Release* it back into the dish.
6. *Turn* the stove *off*.

In order to show that the skills built using our method can be directly combined with a task planner, before running the experiment a STRIPS planner with PDDL-like problem specification, called PyDDL[5], was executed to compute the action order needed to solve the task. Then, the skills were built and associated to each action. Once the skills were available, a task manager executed them based on the action order and the object states. The task manager was also in charge of changing the colors of the screen representing the different object states.

[5]https://github.com/garydoranjr/pyddl


A
next_openness
distance
inclination
orientation
grasped
effect
move_dist
move_orien
move_inclin
Coarse-grained BN (left) and fine-grained BN (right) for the *push* action
B
Coarse-grained BN (left) and fine-grained BN (right) learned for the *grasp* action

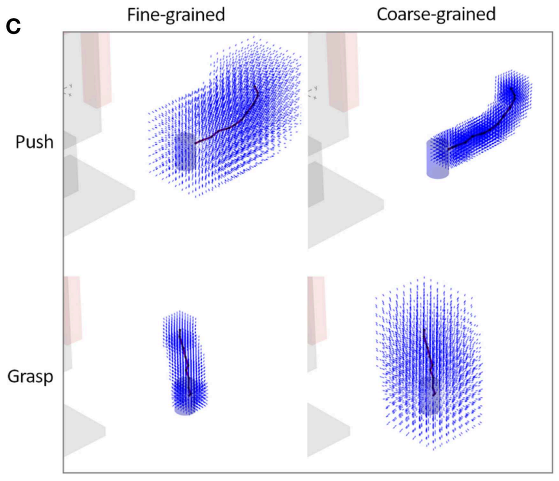

C
Fine-grained
Coarse-grained
Push
Grasp
Examples of vicinities computed using the fine-grained and coarse-grained BNs

FIGURE 5 | (A,B) BNs obtained from the demonstration to *push* and *grasp* an object. **(C)** Corresponding vicinities computed from the same demonstrations using the previous BNs.

4.3. Results

Experiment 1

The results obtained for both mazes are depicted in **Figure 8**. In both cases, the robot was able to solve the maze, showing a high precision for the *push* actions. Therefore, the skills built by IS^{2}L can reproduce effects only based on the robot and object positions, i.e., low-level states. Besides, a skill could reproduce different results of the same effect, e.g., pushing different distances an object. Meaning these skills are task-agnostic and they can be used in different tasks.

At the top of the Figure, J shows the actions executed to solve the first maze. These actions are: (A-B) the robot *set* the end-effector *behind* the cylinder, (B-C) the robot *pushed* the cylinder *far*, (C-D) the robot *set* the end-effector *at the left* of the cylinder, (D-E) the robot *pushed* the cylinder to the *right*, (E-F) the robot *set* the end-effector *in front* of the cylinder, (F-G) the robot *pushed* the cylinder *close*, (G-H) the robot *set* the end-effector *at the right* of the cylinder, (F-G) the robot *pushed* the cylinder to the *left*. All the actions accurately reproduced the expected effect, except setting the arm *at the back* (A-B) and *in front* of the cylinder

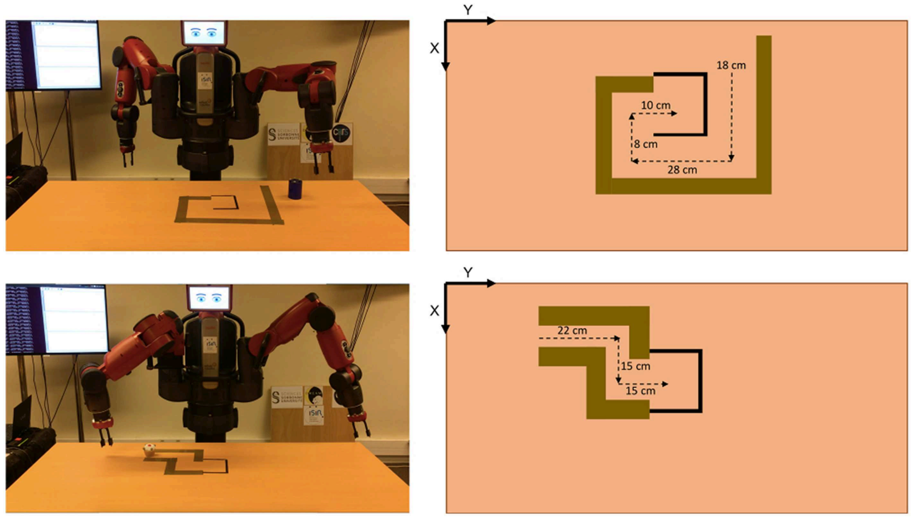


FIGURE 6 | Setup of the mazes used in Experiment 1. At the top, for the first maze, and at the bottom, for the second maze. In both cases, from left to right, the physical setup and the expected distances to *push* the corresponding object in order to solve the maze.

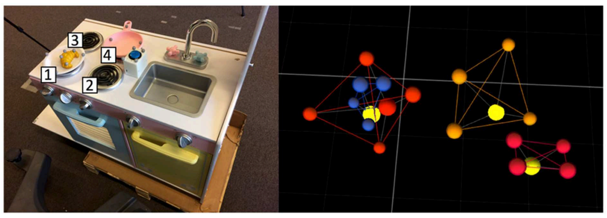


FIGURE 7 | Setup used in Experiments 2 and 3. On the left, an image of the setup from the robot's point of view. On the right, example of the setup acquired by the motion capture system from the same point of view used for the image. The yellow markers represent the position of each object.

(E-F), due to reaching the kinematic limits of the right arm of the robot.

At the bottom of the Figure, F shows the actions executed to solve the second maze. These actions are: (A) the robot *pushed* the cylinder to the *right*, (A-B) the robot *set* the end-effector *at the back* of the cylinder, (B-C) the robot *pushed* the cylinder *far*, (C-D) the robot *set* the end-effector *at the right* of the cylinder, (D-E) the robot *pushed* the cylinder to the *right*, a different distance than A. Similarly, the less accurate actions (A and B) were those reaching the kinematic limits of the robot's arm.

Experiment 2

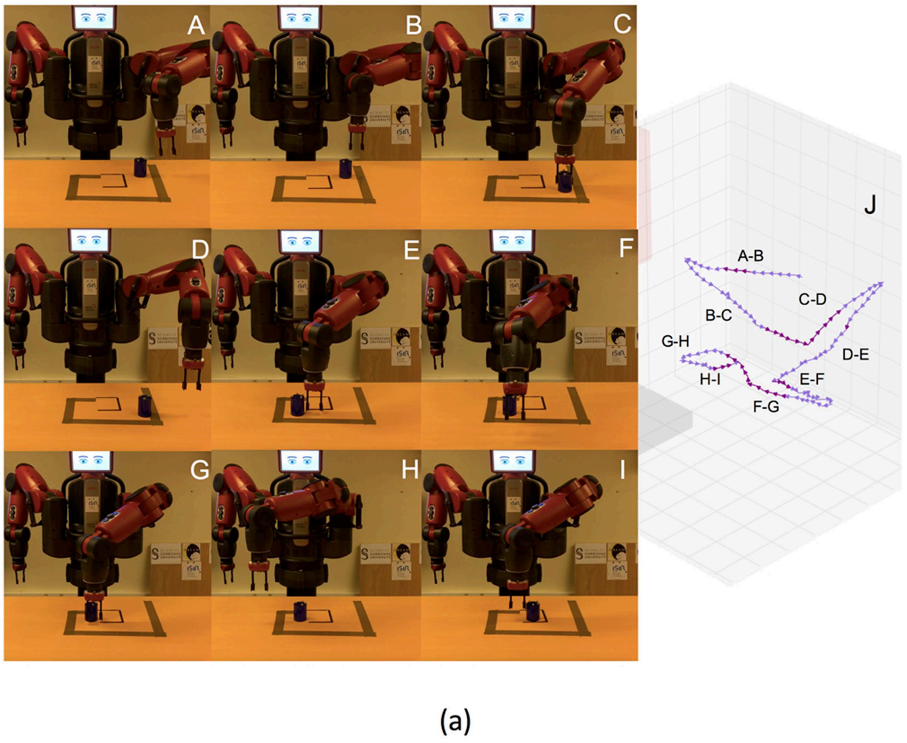
In both tests the *grasp* and *release* actions reproduced the expected effects using both high-level and low-level states. Besides, the skills were robust to the spatio-termporal perturbations, adapting the ongoing actions to the new object and end-effector positions.

Figure 9 shows the trajectories generated in the tests. At the top, the actions and the perturbations for the first test: (A-B) the robot tried to *grasp* the croissant, but its position changed from stove 3 to stove 1, (B-C) the robot adapted the actions and *grasped* it, (C-D) the robot executed the *release* action the croissant into the pan, (D-E) the pan position changed from the stove 4 to the stove 1, and the robot adapted its action, (E-F) the robot *released* the croissant in the pan. The action A-B shows a curve of the action from the end-effector random initial position toward the croissant position, until the latter changes, and a brusque change of direction appears. Similarly, the action C-D follows a trajectory from the croissant position to the pan position. Then, there is an abrupt change in the action direction when the pan position is changed.

At the bottom of the Figure, the actions and the perturbations for the second test: (A-B) from a random initial position over the kitchen the end-effector moved toward the croissant position until its moved from stove 1 to the stove 3, (B-D) the robot action adapted to the new position moving the end-effector far from the robot until the end-effector position was moved farthest than the croissant position, (D-E) the end-effector moved back to the croissant position until the croissant was moved to the stove 4, (E-H) the action again adapted moving toward the right until the end-effector was moved closed to the stove 1, (H-K) again the action adapted to the new end-effector position and moved toward the stove 3 whereas the end-effector was located on top of it, (K-L) finally the croissant is *grasped*. The trajectories of the arm are depicted, where the external changes in the position of the end-effector are identified as long orange arrows. It is very difficult to differentiate the actions due to the high number of changes produced.

Experiment 3

Figure 11 shows the number of effects reproduced in 10 runs of the experiment. Six runs completely reproduced all the effects to solve the task, showing that the skills were able to solve a multi-step task in a realistic environment simultaneously using



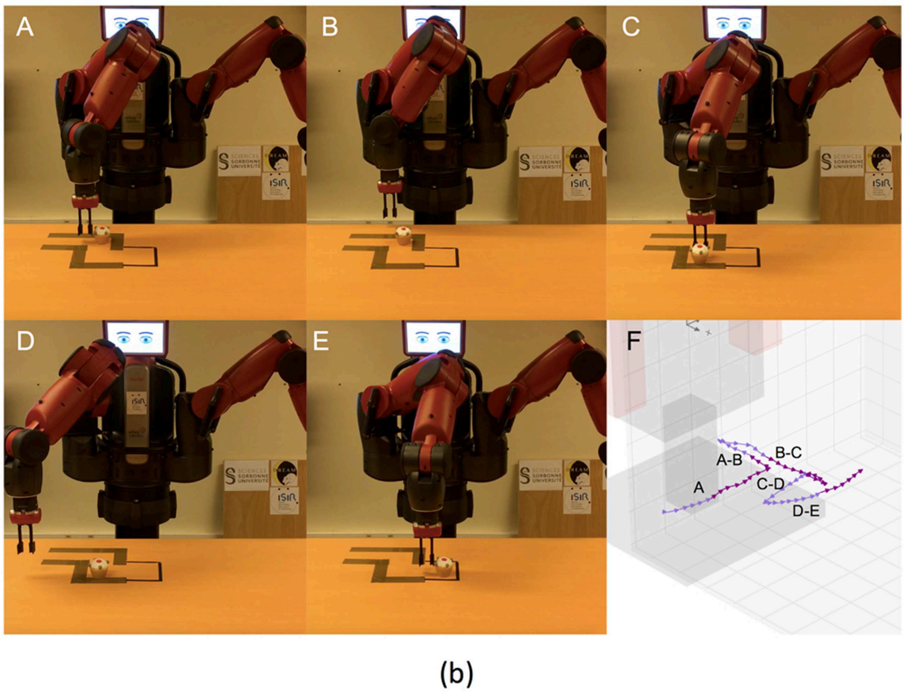
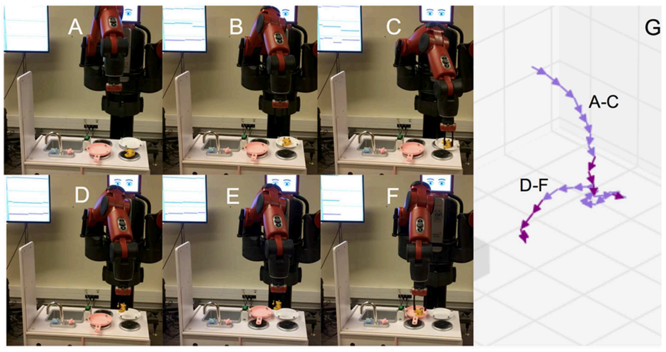
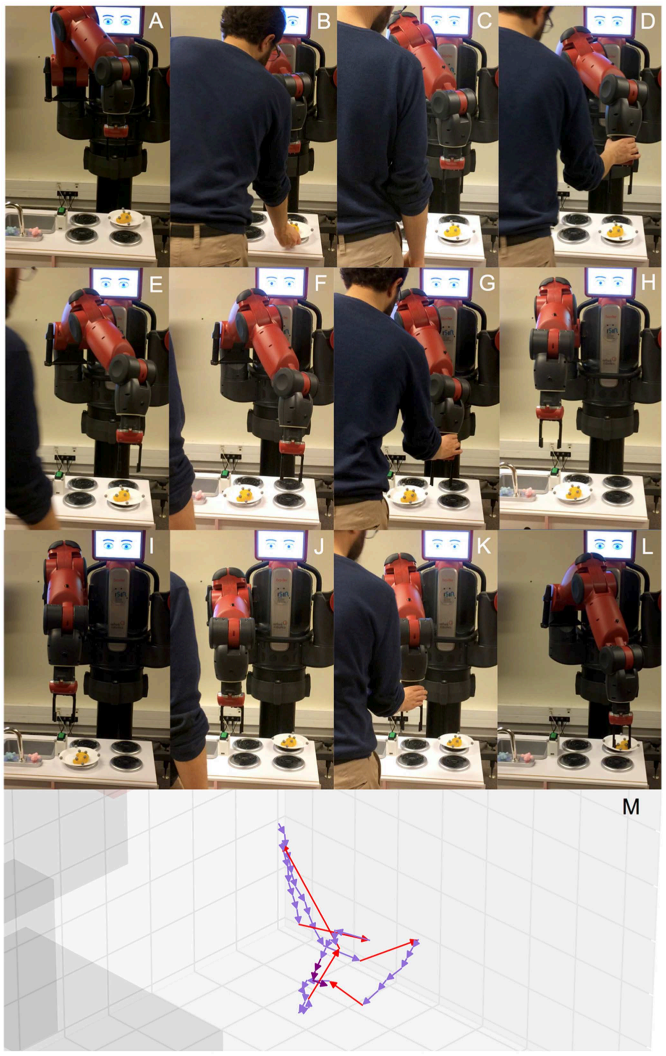


FIGURE 8 | (a) Actions solving the first maze. From A to I, screenshots of the execution of the task. Finally, in J, virtual representation of the actions executed. **(b)** Actions solving the second maze. From A to E, screenshots of the execution of the task. Finally, in F, virtual representation of the actions executed.

the low-level and high-level states of both the robots and the different objects within the scenario. Three times the robot was not able to properly *release* the croissant from the pan to the dish. The *release* actions mainly failed because these actions did not move the end-effector high enough and the markers of the croissant and the pan touched each other, displacing the pan



(a)



(b)

FIGURE 9 | (a) Actions of the first spatio-temporal test, composed of *grasping* the croissant and *releasing* it into the pan. In this case, the spatial perturbation consists in changing the position of the pan during the *release* action. **(b)** Actions of the second spatio-temporal test, in which the robot tries to *grasp* the croissant. Both spatial and temporal perturbations are present, changing the position of the croissant, and externally moving the robot's end-effector, respectively. The orange arrows represent externally generated long movements of the end-effector. At the right side of the robot there is a screen showing the *distance*, *orientation* and *inclination* of the object w.r.t. the robot end-effector.

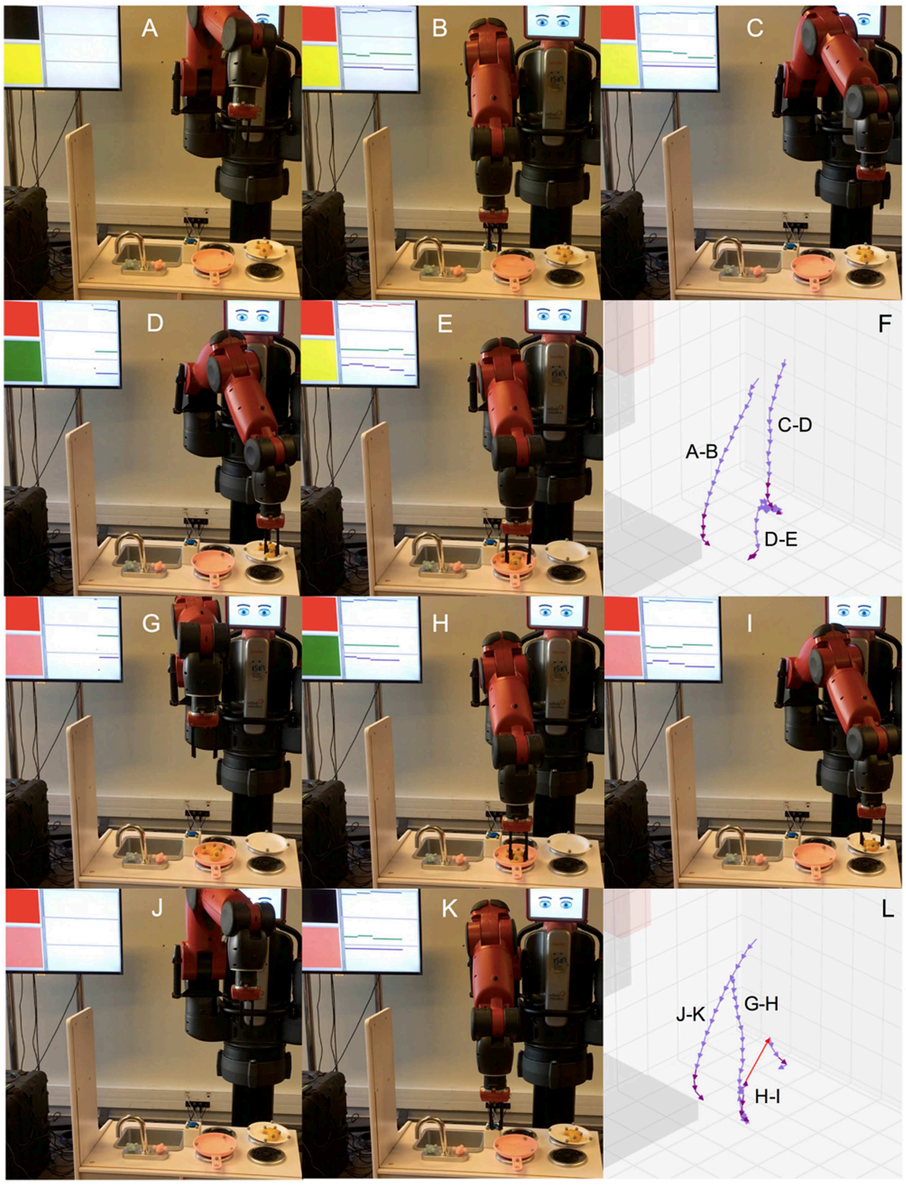


FIGURE 10 | Actions of a successful execution of the task heating the croissant. From A to E, screenshots *pushing* the button, *grasping* the croissant, and *putting it* into the pan. In F, virtual representation of these actions. From G to K, screenshots *grasping* again the croissant, *putting it* back into the dish, and pressing the button. In L, virtual representation of these actions.

and the dish, sometimes making the croissant fall from the end-effector. In one occasion the robot *grasped* the croissant from one of its extremes and this felt back to the dish. **Figure 10** shows the actions of a successful execution of the task heating the croissant: (A-B) from a random initial position the robot *pressed* the button turning the stove 3 on, (C-D) from another random position the robot *grasped* the croissant, (D-E) the croissant was *released* in the pan, (G-H) from another random initial position the *grasp* action is executed once the croissant is hot, i.e., when the color in the screen changes from yellow to salmon, (I-J) the croissant is *released* back in the dish, (J-K) from another initial position the stove is turn off after the button is *pressed*. In general, the actions *pressing* the button were quite accurate. Also, the *grasping* actions were quite robust. The first *release* action (D-E) initially moved the end-effector up a high distance from the dish position, avoiding touching the dish markers. However, in the second *release* action, from the pan to the dish, the movements were lower, producing the objects to touch each other. New demonstrations showing the second *release* action with a higher height would generate a higher release action, importantly improving the success ratio of the experiment.

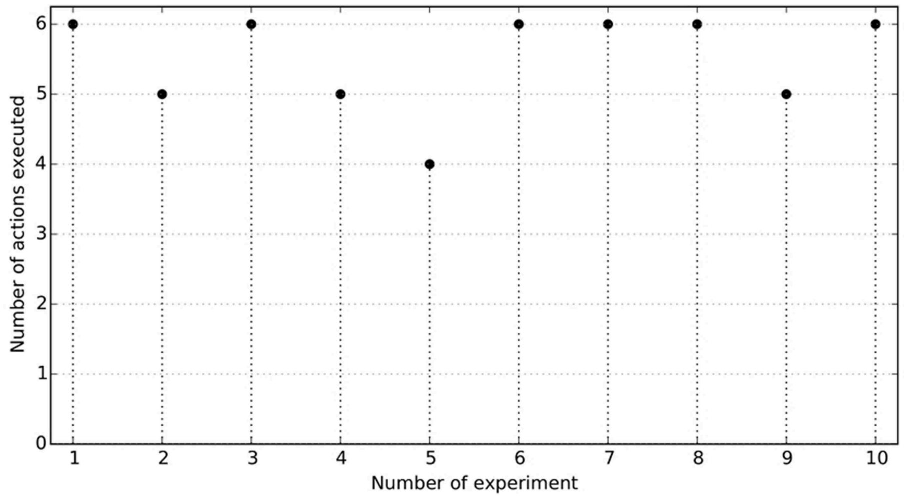


FIGURE 11 | Results of 10 runs of the Experiment 3. The horizontal axis represents the number of action successfully executed of those listed in Experiment 3 (see section 4). A run is successful if the 6 actions are executed reproducing the expected effects.

It is relevant to mention that just after the D-E action, and although the croissant was cold, we forced the task planner to execute the *grasping* action G-H. However, the skill was not able to infer any movement because it was built with the croissant temperature state as *hot*. Few seconds later when this state was reached the *grasp* action started.

5. DISCUSSION AND CONCLUSIONS

In the current paper we introduced a method named Interaction State-based Skill Learning (IS^{2}L) that builds skills to reproduce effects on objects in realistic environments. These environments are three-dimensional and dynamic, i.e., the object states can change at any moment independently of the robot actions. Solving a task in these environments requires the use of complex actions, i.e., pick-and-place an object, that action selection also implies abstract states, e.g., an object is hot or grasped, and actions must be continuous and must adapt to changes in the environment. Therefore, a skill built with our method generates continuous actions that adapt to spatio-temporal perturbation, i.e., it generates in a closed-loop a sequence of movements of the robot's end-effector that adapts to changes of the object position. The skill was implemented as a Bayesian Network (BN). In our previous work (Maestre et al., 2017) we identified a task-agnostic BN structure useful for the action generation. Therefore, building the skill consists in learning the Conditional Probabilistic Distributions (CPDs) of the BN.

Before building the skill the experiment designer creates a dataset of one or more kinesthetic demonstrations of robot-object interactions producing an effect on an object. An interaction is represented as a sequence of high-level and low-level states and the next robot movement to perform at different instants of time. This dataset is used to learn the CPDs allowing the BN to infer the next robot movement for some specific high-level and low-level states. The inference of the this movement is inspired by the affordances action selection. Once this dataset is available the skill building starts, composing two processes: first, the demonstrated interactions are transformed into a repertoire of blocks of information, which represent the previous relationship between the high-level and low-level states and the robot movement. This repertoire is augmented with new relationships to make actions robust to perturbations, using a dynamical system called diffeomorphism. The BNs use discrete values for this relationship, and thus the augmented repertoire of blocks is discretized. In the second process, once this discrete repertoire is available a skill is built, i.e., the CPDs of the BN are learned. This skill infers discrete movements that are afterwards transformed into continuous movements using some simple heuristics.

The main contribution of this paper is a combination of the main features of dynamical systems and affordances in a unique method to build skills that solve tasks in realistic scenarios. More precisely, combining the low-level movement generation of the dynamical systems, to adapt to local perturbations, with the next movement selection simultaneously based on high-level and low-level states.

It is relevant to remark that for each experiment two BNs with different levels of accuracy were learned: a coarse-grained BN inferring bigger movements approaching the end-effector to the object, and a fine-grained BN inferring small and more accurate movements. This design decision was made to speed up the action execution, although only using the fine-grained BN would have been enough to generate the action. The minimal object distance to switch from the coarse-grained BN to the fine-grained one was set from experience, although it could be automatically identified based on a trial-and-error approach executing the same action few times (under similar conditions) and analyzing the quality of the obtained effect.

The learning of the BNs was fast and straightforward. However, depending on the size of the CPDs the movement inference was slow, and thus the movements of the robot although smooth were not as realistic as expected. Each BN structure is generated based on the corresponding dataset, and it represents different dependencies between the nodes based on the data available. The BN structures of the *push* and *grasp* skills identify the dependencies of the movement to perform with respect to the nodes describing the robot and object state before the execution of the movement, i.e., the relation state and the object being grasped. However, the dependency related to the *openness* of the gripper is only identified for the *grasp* skill, due to for the *push* skill the gripper remains closed.

Defining an action as a sequence of pairwise movements generated quite smooth trajectories. The interaction demonstrations were simple to generate, and to combine, providing a simple and flexible way of creating the initial dataset. The execution of these actions may produce some robot-object relation states unseen during the demonstrations, due to the noise generated by the robot joints. The method is robust to these situations thanks to the data augmentation, generating many different relation states around the demonstrations.

Although the quantity of a priori information needed to build the skills may look relevant, we have explained that in most cases the same values are useful for tasks sharing common abstract features, i.e., a gripper interacting with an object. Therefore, once a correct value is found for a variable this can be used for many different experiments.

A Baxter robot performed three experiments solving tasks of increasing complexity, using both low-level and high-level states. These experiments demonstrated that the method is able to generate skills useful for task solving in realistic environments. However, it is relevant to mention some aspects that can limit the scaling up of our method to more complex scenarios. The size of the CPDs grows exponentially when new states become relevant for a task, because of the *curse of dimensionality*. For example, if grasping objects of specific size, color and/or orientation. Similarly, performing tasks involving more than one object would generate the same dimensionality issue. This constraint has been already solved in Goncalves et al. (2014) reducing the dimensionality of the information provided to the BN using the Principal Component Analysis (PCA) technique. This approach would allow the BN to handle more information. However, the use of PCA or other dimensionality reduction techniques could complicate the identification of the proper BN structure. Another possible limitation is related to the complexity of the tasks. It would be necessary to test if tasks requiring high accuracy, e.g., putting a key in a keyhole and turning it, could be accomplished with the current task-agnostic parametrization of the method. Mainly with the proposed discretization configuration. Possibly new heuristics for the distance discretization would be necessary to generate more accurate robot movements.

Some possible improvements to the method would be to exploit all the information the dynamical system provides about the next movement, i.e., the *orientation*, *velocity* and *acceleration*. Currently, only the orientation is used by our method with a constant velocity. Extending the method to use the velocity and acceleration would result in more complex actions, e.g., *poke*. Also w.r.t. the dynamical system, adding areas to avoid to the generated vector fields, called *repellers*, would provide to the method an obstacle avoidance capacity, allowing the use of the method in more realistic environments.

AUTHOR CONTRIBUTIONS

CM was involved in the conception, design, and coding of the method. SD and CG are involved in the conception and design of the method. GM was involved in the coding of the method. CM, SD, CG, and GM wrote the paper.

FUNDING

This research was sponsored in part by the DREAM project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 640891. This work was performed within the Labex SMART (ANR-11-LABX-65) supported by French state funds managed by the ANR within the Investissements d'Avenir programme under reference ANR-11-IDEX-0004-02.

REFERENCES

Antunes, A., Jamone, L., Saponaro, G., Bernardino, A., and Ventura, R. (2016). "From human instructions to robot actions: formulation of goals, affordances and probabilistic planning," in *Proceedings - IEEE International Conference on Robotics and Automation* (Stockholm), 5449–5454.

Atkeson, C. G., Babu, B. P. W., Banerjee, N., Berenson, D., Bove, C. P., Cui, X., et al. (2018). "What Happened at the DARPA Robotics Challenge Finals," in *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*, eds M. Spenko, S. Buerger, and K. Iagnemma (Cham: Springer), 667–684. doi: 10.1007/978-3-319-74666-1_17

Billard, A. G., and Calinon, S. (2016). Handbook of robotics chapter 59 : robot programming by demonstration. *Robotics* 48, 1371–1394. doi: 10.1007/978-3-540-30301-5_60

Calinon, S. (2016). A tutorial on task-parameterized movement learning and retrieval. *Intell. Ser. Robot.* 9, 1–29. doi: 10.1007/s11370-015-0187-9

Calinon, S., D'halluin, F., Sauser, E. L., Caldwell, D. G., and Billard, A. G. (2010). A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation. *IEEE Robot. Automat. Mag.* 17, 44–54. doi: 10.1109/MRA.2010.936947

Calinon, S., Guenter, F., and Billard, A. G. (2007). On learning, representing, and generalizing a task in a humanoid robot. *IEEE Trans. Syst. Man Cybern. B Cybern.* 37, 286–298. doi: 10.1109/TSMCB.2006.886952

Calinon, S., Pistillo, A., and Caldwell, D. G. (2011). "Encoding the time and space constraints of a task in explicit-duration hidden Markov model," in *IEEE International Conference on Intelligent Robots and Systems* (San Francisco, CA), 3413–3418.

Chavez-Garcia, R. O., Andries, M., Luce-Vayrac, P., and Chatila, R. (2017). "Discovering and manipulating affordances," in *International Symposium on Experimental Robotics* (Tokyo), 679–691.

Chavez-Garcia, R. O., Luce-Vayrac, P., and Chatila, R. (2016). "Discovering affordances through perception and manipulation," in *IEEE International Conference on Intelligent Robots and Systems* (Deajeon), 3959–3964.

Dearden, A., and Demiris, Y. (2005). "Learning forward models for robots," in *IJCAI International Joint Conference on Artificial Intelligence* (Edinburgh), 1440–1445.

Dehban, A., Jamone, L., and Kampff, A. R. (2017). "A deep probabilistic framework for heterogeneous self-supervised learning of affordances," in *Humanoids 2017* (Birmingham).

Dehban, A., Jamone, L., Kampff, A. R., and Santos-Victor, J. (2016). "Denoising auto-encoders for learning of objects and tools affordances in continuous space," in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (Stockholm), 4866–4871. doi: 10.1109/ICRA.2016.7487691

Demiris, Y., and Dearden, A. (2005). "From motor babbling to hierarchical learning by imitation: a robot developmental pathway," in *International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems* (Nara), 31–37.

Ebert, F., Finn, C., Lee, A. X., and Levine, S. (2017). "Self-supervised visual planning with temporal skip connections," in *1st Annual Conference on Robot Learning, CoRL 2017* (Mountain View, CA).

Finn, C., and Levine, S. (2017). "Deep visual foresight for planning robot motion," in *Proceedings - IEEE International Conference on Robotics and Automation* (Singapore), 2786–2793.

Finn, C., Goodfellow, I., and Levine, S. (2016). "Unsupervised learning for physical interaction through video prediction," in *NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems* (Barcelona: Curran Associates, Inc.), 64–72.

Fitzpatrick, P., and Metta, G. (2003). Grounding vision through experimental manipulation. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* 361, 2165–2185. doi: 10.1098/rsta.2003.1251

Fitzpatrick, P., Metta, G., Natale, L., Rao, S., and Sandini, G. (2003). "Learning about objects through action-initial steps towards artificial cognition," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)* (Taipei), 3140–3145.

Gibson, E. J. (2000). Perceptual learning in development: some basic concepts. *Ecol. Psychol.* 12, 295–302. doi: 10.1207/S15326969ECO1204_04

Gibson, E. J. (2003). The world is so full of a number of things: on specification and perceptual learning. *Ecol. Psychol.* 15, 283–287. doi: 10.1207/s15326969eco1504_3

Gibson, J. J. (1966). *The Senses Considered as Perceptual Systems*. Boston, MA: Houghton Mifflin.

Gibson, J. J. (1986). *The Ecological Approach to Visual Perception*. Hillsdale, NJ: L. Erlbaum. 127–136.

Goncalves, A., Abrantes, J., Saponaro, G., Jamone, L., and Bernardino, A. (2014). "Learning intermediate object affordances: towards the development of a tool concept," in *IEEE ICDL-EPIROB 2014 - 4th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics* (Genoa), 482–488. doi: 10.1109/DEVLRN.2014.6983027

Gribovskaya, E., Khansari-Zadeh, S. M., and Billard, A. G. (2011). Learning non-linear multivariate dynamics of motion in robotic manipulators. *Int. J. Robot. Res.* 30, 80–117. doi: 10.1177/0278364910376251

Hangl, S., Ugur, E., Szedmak, S., and Piater, J. (2016). "Robotic playing for hierarchical complex skill learning," in *IEEE International Conference on Intelligent Robots and Systems* (Deajeon), 2799–2804.

Hart, S., Grupen, R., and Jensen, D. (2005). "A relational representation for procedural task knowledge," in *Proceedings of the 20th National Conference on Artificial Intelligence* (Pittsburgh), 1280–1285.

Hermans, T., Li, F., Rehg, J. M., and Bobick, A. F. (2013). "Learning contact locations for pushing and orienting unknown objects," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (Atlanta), 435–442.

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9, 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. (2013). Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Comput.* 25, 328–373. doi: 10.1162/NECO_a_00393

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems 15 (NIPS2002)* (Vancouver, BC), 1547–1554.

Jain, R., and Inamura, T. (2011). "Learning of tool affordances for autonomous tool manipulation," in *2011 IEEE/SICE International Symposium on System Integration, SII 2011* (Kyoto), 814–819.

Jain, R., and Inamura, T. (2013). Bayesian learning of tool affordances based on generalization of functional feature to estimate effects of unseen tools. *Artif. Life Robot.* 18, 95–103. doi: 10.1007/s10015-013-0105-1

Jamone, L., Ugur, E., Cangelosi, A., Fadiga, L., Bernardino, A., Piater, J., et al. (2016). Affordances in psychology, neuroscience and robotics: a survey. *IEEE Trans. Cogn. Dev. Syst.* 10, 4–25. doi: 10.1109/TCDS.2016.2594134

Jonschkowski, R., and Brock, O. (2015). Learning state representations with robotic priors. *Auton. Robots* 39, 407–428. doi: 10.1007/s10514-015-9459-7

Jonschkowski, R., Hafner, R., Scholz, J., and Riedmiller, M. (2017). PVEs: position-velocity encoders for unsupervised learning of structured state representations. *arXiv:1705.09805*.

Khansari-Zadeh, S. M., and Billard, A. G. (2011). Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Trans. Robot.* 27, 943–957. doi: 10.1109/TRO.2011.2159412

Khansari-Zadeh, S. M., and Billard, A. G. (2014). Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robot. Auton. Syst.* 62, 752–765. doi: 10.1016/j.robot.2014.03.001

Kim, S., Shukla, A., and Billard, A. G. (2014). Catching objects in flight. *IEEE Trans. Robot.* 30, 1049–1065. doi: 10.1109/TRO.2014.2316022

Kober, J., Muelling, K., Kroemer, O., Lampert, C. H., Scholkopf, B., and Peters, J. (2010). "Movement templates for learning of hitting and batting," in *2010 IEEE International Conference on Robotics and Automation* (Karlsruhe), 853–858.

Kopicki, M., Zurek, S., Stolkin, R., Mörwald, T., and Wyatt, J. (2011). "Learning to predict how rigid objects behave under simple manipulation," in *Proceedings - IEEE International Conference on Robotics and Automation* (Shanghai), 5722–5729.

Kroemer, O., Ugur, E., Oztop, E., and Peters, J. (2012). "A kernel-based approach to direct action perception," in *IEEE International Conference on Robotics and Automation (ICRA)* (St. Paul, MN), 2605–2610.

Krotkov, E. (1995). "Robotic perception of material," in *Proceedings of the IJCAI* (Montreal, QC), 88–94.

Krüger, N., Geib, C., Piater, J., Petrick, R., Steedman, M., Wörgötter, F., et al. (2011). ObjectAction complexes: grounded abstractions of sensorymotor processes. *Robot. Auton. Syst.* 59, 740–757. doi: 10.1016/j.robot.2011.05.009

Kugler, P. N., and Turvey, M. T. (1987). "Information, natural law, and the self-assembly of rhythmic movement," in *Resources for Ecological Psychology* (Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.), 481.

Lopes, M., Melo, F. S., and Montesano, L. (2007). "Affordance-based imitation learning in robots," in *IEEE International Conference on Intelligent Robots and Systems* (San Diego, CA), 1015–1021.

Maestre, C., Mukhtar, G., Gonzales, C., and Doncieux, S. (2017). "Iterative affordance learning with adaptive action generation," in *ICDL-Epirob - International Conference on Development and Learning, Epirob* (Lisbon).

May, S., Klodt, M., Rome, E., and Breithaupt, R. (2007). "GPU-accelerated affordance cueing based on visual attention," in *IEEE International Conference on Intelligent Robots and Systems* (San Diego, CA), 3385–3390.

Metta, G., and Fitzpatrick, P. (2003). Better Vision through Manipulation. *Adapt. Behav.* 11, 109–128. doi: 10.1177/10597123030112004

Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning object affordances: from sensory–motor coordination to imitation. *IEEE Trans. Robot.* 24, 15–26. doi: 10.1109/TRO.2007.914848

Muelling, K., Kober, J., Kroemer, O., and Peters, J. (2013). Learning to select and generalize striking movements in robot table tennis. *Int. J. Robot. Res.* 32, 263–279. doi: 10.1177/0278364912472380

Mugan, J., and Kuipers, B. J. (2012). Autonomous learning of high-level states and actions in continuous environments. *IEEE Trans. Auton. Ment. Dev.* 4, 70–86. doi: 10.1109/TAMD.2011.2160943

Omrčen, D., Ude, A., and Kos, A. (2008). "Learning primitive actions through object exploration," in *2008 8th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2008* (Daejeon), 306–311.

Osório, P., Bernardino, A., Martinez-Cantin, R., and Santos-Victor, J. (2010). "Gaussian mixture models for affordance learning using Bayesian networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Taipei), 4432–4437.

Paraschos, A., Daniel, C., Peters, J., and Neumann, G. (2013). "Probabilistic movement primitives," in *Neural Information Processing Systems*, 1–9.

Paraschos, A., Daniel, C., Peters, J., and Neumann, G. (2017). Using probabilistic movement primitives in robotics. *Autonomous Robots* 42, 529–551.

Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. (2009). “Learning and generalization of motor skills by learning from demonstration,” in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation* (Kobe), 1293–1298.

Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Perrin, N., and Schlehuber-Caissier, P. (2016). Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Syst. Control Lett.* 96, 51–59. doi: 10.1016/j.sysconle.2016.06.018

Ridge, B., Skočaj, D., and Leonardis, A. (2010). “Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems,” in *Proceedings - IEEE International Conference on Robotics and Automation* (Anchorage), 5047–5054.

Sahin, E., Cakmak, M., Dogar, M. R., Ugur, E., and Ucoluk, G. (2007). To afford or not to afford: a new formalization of affordances toward affordance-based robot control. *Adapt. Behav.* 15, 447–472. doi: 10.1177/1059712307084689

Slotine, J.-J. E., and Li, W. (1991). *Applied Nonlinear Control*, Prentice-Hall.

Stoytchev, A. (2005). “Toward learning the binding affordances of objects: a behavior-grounded approach,” in *AAAI Symposium on Developmental Robotics*, 21–23.

Szedmak, S., Ugur, E., and Piater, J. (2014). “Knowledge propagation and relation learning for predicting action effects,” in *IEEE International Conference on Intelligent Robots and Systems* (Chicago, IL: Institute of Electrical and Electronics Engineers Inc.), 623–629.

Ugur, E., Nagai, Y., Sahin, E., and Oztop, E. (2015a). Staged development of robot skills: behavior formation, affordance learning and imitation with motionese. *IEEE Trans. Auton. Ment. Dev.* 7, 119–139. doi: 10.1109/TAMD.2015.2426192

Ugur, E., Oztop, E., and Sahin, E. (2011). Goal emulation and planning in perceptual space using learned affordances. *Robot. Auton. Syst.* 59, 580–595. doi: 10.1016/j.robot.2011.04.005

Ugur, E., and Piater, J. (2015). “Refining discovered symbols with multi-step interaction experience,” in *IEEE-RAS International Conference on Humanoid Robots* (Seoul), 1007–1012.

Ugur, E., Piater, J., Silberman, N., Hoiem, D., Kohli, P., Fergus, R., et al. (2015b). Bottom-up learning of object categories, action effects and logical rules: from continuous manipulative exploration to symbolic planning,” in *IEEE International Conference on Robotics and Automation*, Vol. 4 (Seattle, MA), 1.

Ugur, E., Sahin, E., and Oztop, E. (2009). “Affordance learning from range data for multi-step planning,” in *Proceedings of the 9th International Conference on Epigenetic Robotics, Lund University Cognitive Studies*, eds L. Cañamero, P.-Y. Oudeyer, and C. Balkenius (Venice: Lund University), 177–184.

Ugur, E., Sahin, E., and Oztop, E. (2012). “Self-discovery of motor primitives and learning grasp affordances,” in *IEEE IROS* (Vilamoura).

Warren, W. H. (1988). Action modes and laws of control for the visual guidance of action. *Adv. Psychol.* 50, 339–379. doi: 10.1016/S0166-4115(08)62564-9

Wong, J. M. (2016). Towards lifelong self-supervision: a deep learning direction for robotics. *arXiv:1611.00201*.

Zech, P., Haller, S., Lakani, S. R., Ridge, B., Ugur, E., and Piater, J. (2017). Computationalmodels of affordance in robotics: a taxonomy and systematic classification. *Adapt. Behav.* 25, 235–271. doi: 10.1177/1059712317726357

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 26 July 2019
doi: 10.3389/fnbot.2019.00045

An Embodied Agent Learning Affordances With Intrinsic Motivations and Solving Extrinsic Tasks With Attention and One-Step Planning

Gianluca Baldassarre[1]*, William Lord[2], Giovanni Granato[1] and Vieri Giuliano Santucci[1]

[1] Laboratory of Computational Embodied Neuroscience, Institute of Cognitive Sciences and Technologies, National Research Council of Italy, Rome, Italy, [2] School of Engineering Sciences, KTH Royal Institute of Technology, Stockholm, Sweden

We propose an architecture for the open-ended learning and control of embodied agents. The architecture learns action affordances and forward models based on intrinsic motivations and can later use the acquired knowledge to solve extrinsic tasks by decomposing them into sub-tasks, each solved with one-step planning. An affordance is here operationalized as the agent's estimate of the probability of success of an action performed on a given object. The focus of the work is on the overall architecture while single sensorimotor components are simplified. A key element of the architecture is the use of "active vision" that plays two functions, namely to focus on single objects and to factorize visual information into the object appearance and object position. These processes serve both the acquisition and use of object-related affordances, and the decomposition of extrinsic goals (tasks) into multiple sub-goals (sub-tasks). The architecture gives novel contributions on three problems: (a) the learning of affordances based on intrinsic motivations; (b) the use of active vision to decompose complex extrinsic tasks; (c) the possible role of affordances within planning systems endowed with models of the world. The architecture is tested in a simulated stylized 2D scenario in which objects need to be moved or "manipulated" in order to accomplish new desired overall configurations of the objects (extrinsic goals). The results show the utility of using intrinsic motivations to support affordance learning; the utility of active vision to solve composite tasks; and the possible utility of affordances for solving utility-based planning problems.

Keywords: open-ended learning, intrinsic motivations, affordance learning, goal-based planning, utility-based planning, active-vision, attention

OPEN ACCESS

Edited by:
Alex Pitti,
Université de Cergy-Pontoise, France

Reviewed by:
Frederic Alexandre,
Inria Bordeaux - Sud-Ouest Research Centre, France
Benoît Girard,
Centre National de la Recherche Scientifique (CNRS), France

****Correspondence:***
Gianluca Baldassarre
gianluca.baldassarre@istc.cnr.it

Received: *28 September 2018*
Accepted: *11 June 2019*
Published: *26 July 2019*

Citation:
Baldassarre G, Lord W, Granato G and Santucci VG (2019) An Embodied Agent Learning Affordances With Intrinsic Motivations and Solving Extrinsic Tasks With Attention and One-Step Planning. Front. Neurorobot. 13:45. doi: 10.3389/fnbot.2019.00045

1. INTRODUCTION

This work proposes an architecture for the control and learning of embodied agents. The architecture has been developed within an open-ended learning context. **Figure 1** shows a typical scenario used in such a context[1]: the scenario is used here to test the proposed architecture. The general structure of the scenario involves two phases (Baldassarre, 2011; Seepanomwan et al., 2017):

[1]This and more complex versions of the scenario involving autonomous robots have been developed within the EU funded project "GOAL-Robots – Goal-based Open-ended learning Autonomous Robots," www.goal-robots.eu.

(a) a first *intrinsic motivation phase* where the agent is not given any task and should freely explore the environment to autonomously acquire as much general-purpose knowledge as possible; (b) a second *extrinsic motivation phase* where the agent has to solve one or more tasks assigned externally within the same environment (*extrinsic tasks*). Importantly, the extrinsic phase can furnish an objective measure of the quality of the algorithms used by the agent to autonomously learn during the intrinsic phase. In the intrinsic phase of the specific scenario used here, the agent can perceive objects and explore and learn the effects of certain pre-wired actions (e.g., "move in space" or "change object color"). In the extrinsic phase, the agent is required to use the knowledge acquired in the intrinsic phase to solve extrinsic tasks: first the agent has to memorize the state of some objects set in a certain configuration (*goal*; notice how this is a handy way to allow the agent to store the goal in a format suitable for its processes); then the objects are "shuffled" into a different state ("initial state"); last the agent has to bring the objects back to the goal state.

Facing the challenges posed by the scenario requires different functions. The functions used by the architecture proposed here are summarized in **Figure 2**. The figure shows that during the intrinsic phase the architecture uses intrinsic motivations to learn action affordances and forward models, and during the extrinsic phase it uses affordances and forward models to plan and solve the extrinsic tasks. Importantly, in both phases active vision allows the agent to focus on a single object per time, in particular to elicit object-centered intrinsic motivations, to learn or activate the affordances and the forward models related to specific objects, and to parse the extrinsic goal into simpler sub-goals each achievable with 1-step planning. These processes are now considered more in detail. For each process we now highlight the relevant concepts and literature and introduce the open problems faced here (section 4 compares the architecture with other specific models proposed in the literature). We then illustrate the three main contributions of the work.

1.1. Links to the Literature and Open Problems

The proposed architecture has been developed within the area of developmental and autonomous robotics called *open-ended learning* (Thrun and Mitchell, 1995; Weng et al., 2001). Open-ended learning processes allow robots to acquire knowledge (e.g., goals, action policies, forward models, and inverse models, etc.) in an incremental fashion by interacting with the environment. These learning processes are strongly inspired by the exploration processes seen in animals, in particular in humans and especially in children (Asada et al., 2001; Lungarella et al., 2003). Although open-ended learning processes can involve both social and individual mechanisms (Baldassarre and Mirolli, 2013a), here we only focus on individual learning processes supporting an autonomous acquisition of knowledge.

A central concept in open-ended learning is the one of *intrinsic motivations* (IMs). These are mechanisms for driving autonomous learning (White, 1959; Ryan and Deci, 2000; Oudeyer and Kaplan, 2007; Baldassarre and Mirolli, 2013a). The utility and adaptive function of IMs reside in that they can produce learning signals, or trigger the performance of behaviors, to drive the acquisition of knowledge and skills that become useful only in later stages with respect to the time in which they are acquired (Baldassarre, 2011). There are various IM mechanisms, some based on the *novelty* or *surprise* of the stimuli (Baldassarre and Mirolli, 2013b; Barto et al., 2013), and others based on the agent's *competence*, i.e., its capacity to successfully accomplish a desired outcome (Mirolli and Baldassarre, 2013).

While various works use IMs as a means to directly guide the autonomous learning of skills (e.g., Schmidhuber, 1991b; Oudeyer et al., 2007), recently they have been used in connection to goals. In particular, surprise or novelty IMs can be used to generate goals, and the goal accomplishment rate can be used to measure competence (Barto et al., 2004; Santucci et al., 2012, 2016; Baranes and Oudeyer, 2013). By *goal* here we refer to an internal representation of the agent having the following properties (cf. Fikes and Nilsson, 1972; Bratman, 1987): (a) the representation refers to a possible future state/state-trajectory (or set of states/state-trajectories) of the world; (b) the representation can be re-activated on the basis of internal processes in the absence of that state/state-trajectory in the world; (c) if the agent "activates"/"commits" to the goal, the goal motivates the performance of some behaviors, in particular the performance of behaviors that tend to push the environment toward the goal state; (d) when the environment reaches (or is close to) the goal state, learning signals or motivations to act in a certain way can be generated; (e) a sub-goal is a goal that is not pursued *per se* but as a means to achieve a desired "final" goal. The architecture presented here, as usually done in the robotic literature on affordances (see below), assumes the agent is given a set of actions and the capacity to recognize if the performance of those actions leads to their desired effects (goals): the challenge for the robot is indeed to use intrinsic motivations to learn which actions can be successfully accomplished on which objects (object affordances).

Much research on open-ended learning has focused on the autonomous acquisition of knowledge during "intrinsically motivated" phases. On the other side, only a few works (e.g., Schembri et al., 2007; Santucci et al., 2014; Seepanomwan et al., 2017) have focused on how such knowledge can be exploited later to solve "extrinsic tasks," namely tasks that produce a material utility to the agent (or to its user in the case of robots, Baldassarre, 2011). The work presented here focuses on problems involving both the intrinsic and extrinsic phases. Although the intrinsic and extrinsic learning processes are often intermixed in realistic situations, separating them can help to clarify problems and to develop algorithms, most notably to use the performance in the extrinsic phase to measure the quality of the knowledge autonomously acquired in the intrinsic phase.

In the extrinsic phase, we consider a test that requires solving a complex task formed by multiple sub-tasks each involving a specific object. The reason why we focus on these types of complex tasks is that: (a) they are involved in most sensorimotor non-navigation robotics scenarios requiring object-manipulation; (b) active perception (see below) can be extremely useful to tackle these scenarios. A possible strategy to solve complex tasks is based on *planning* (Ghallab et al.,

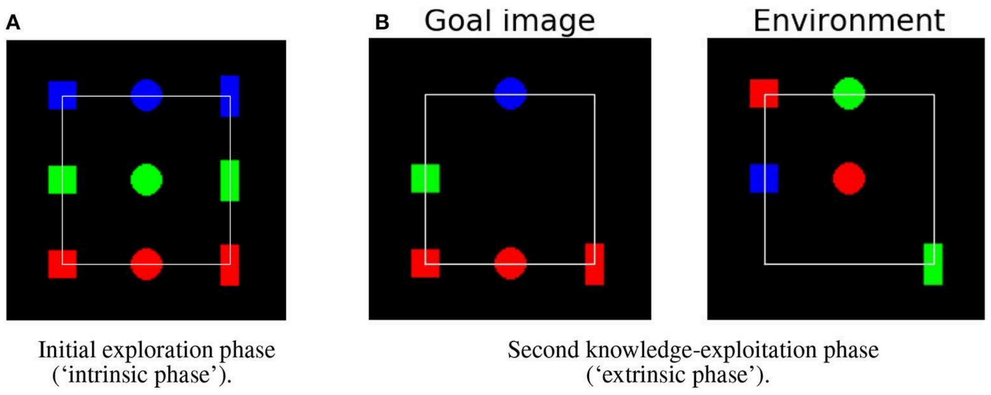

A
B Goal image
Environment
Initial exploration phase ('intrinsic phase').
Second knowledge-exploitation phase ('extrinsic phase').

FIGURE 1 | Example scenario used to test the open-ended learning architecture proposed here. **(A)** Intrinsic phase: initial configuration of the environment containing nine 2D "objects." Each object has a certain position and color (the "object state"). In this phase the agent can autonomously explore the objects for a certain time to acquire as much knowledge and skills as possible. **(B)** Extrinsic phase: example of a task that the agent has to solve, requiring to either move objects or change their colors to bring the "environment" to the state of the "goal image"; the agent has to be able to do this on the basis of the knowledge and skills acquired during the intrinsic phase. Note that in these images the objects are set on the vertices of a 3 × 3 regular grid, although during exploration and extrinsic-task solving they can occupy any non-overlapping position with their center within the white square frame.

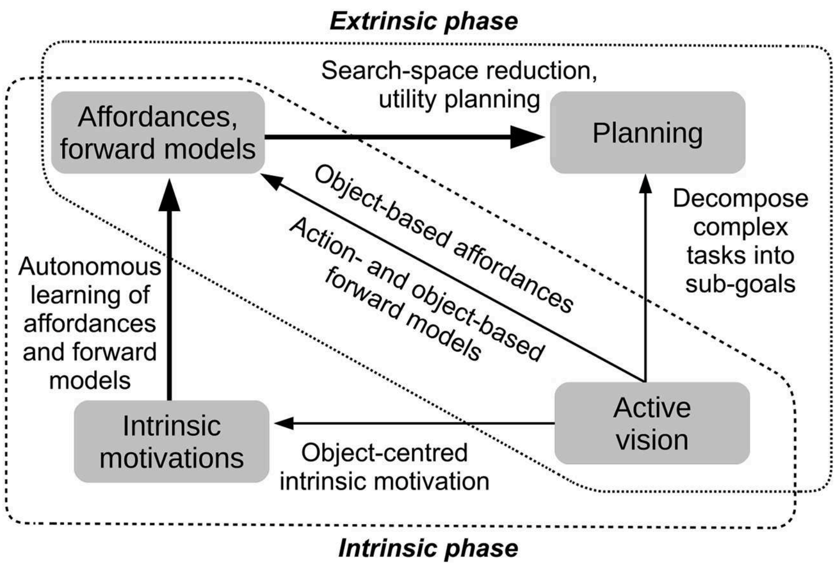

Extrinsic phase
Search-space reduction, utility planning
Affordances, forward models
Planning
Object-based affordances
Action- and object-based forward models
Decompose complex tasks into sub-goals
Autonomous learning of affordances and forward models
Intrinsic motivations
Active vision
Object-centred intrinsic motivation
Intrinsic phase

FIGURE 2 | Main functions (gray boxes) and their relations (arrows with labels) implemented by the architecture proposed here. The dashed and dotted frames contain the main functions used in the intrinsic phase and extrinsic phase, respectively. During the intrinsic phase *intrinsic motivations* support the autonomous learning of *affordances and forward models*; during the extrinsic phase, *affordances and forward models* support *planning* to accomplish extrinsic tasks (bold arrows). Importantly, in the two phases *active vision* supports these processes (thin arrows): to generate intrinsic motivations linked to objects; to learn affordances and forward models related to objects; and to plan based on parsing the goal into object-related sub-goals.

2004), directed to assemble sequences of sub-goals/skills leading to accomplishing the overall complex goal. In this work we focus on a class of tasks where the sub-tasks are (cf. Korf, 1985; Russell and Norvig, 2016): (a) *independent* between them, meaning that the accomplishment of one of them does not require the previous accomplishment of another; (b) *serializable*, meaning that the accomplishment of each sub-goal does not violate other already accomplished sub-goals; (c) can be achieved with a single action available to the agent. An example of such a task in the domain considered here, involving two sub-goals, is: "turn the red circle into a blue circle and move the red square to a desired x-y position." This type of task can be solved with repeated single-step planning processes. The use of this simple class of problems allows us to focus on the issues illustrated below.

In order to study the relations between affordances (see below) and planning, we focus here on two types of planning strategies investigated in the literature on planning (Ghallab et al., 2004; Russell and Norvig, 2016). The first type involves *goal-based planners* that have to decide which actions to perform to accomplish a desired future condition (goal). The second type involves *utility-based planners* that have to decide between alternative conflicting goals having a different desirability and pursued in uncertain conditions (here the stochasticity of the environment is due to the fact that actions succeed only with a certain probability).

As mentioned above, an important dimension of the focus of this work concerns *affordances*. This concept was first proposed within the psychological literature to refer to the actions that a given condition or object "offers" to an agent given its current body state (Gibson, 1979). The concept has been later used in the developmental-robotics literature to refer to the dyadic relational concept for which a certain condition or *object* allows performing a certain *action* on it (Stoytchev, 2005; Sweeney and Grupen, 2007). The interest of affordances for autonomous robotics, as we will also show here, resides in the fact that they can represent a simple and efficient mechanism to rapidly decide which actions can be performed, with some potential utility, on the objects available in the environment.

In the literature, the concept of affordance has occasionally been broadened to refer to various elements relevant for planning. For example, affordances have been associated with three functions linking three critical elements of behavior (Montesano et al., 2008; Ugur et al., 2011): an *Object* (O), the *Action* (A) to perform on it, and the resulting *Effect* (E). The three functions, denoted here as F, C and I, get as input two of the three elements and return the remaining one as output: (a) forward model, $E = F(O, A)$; (b) relevance, $O = C(A, E)$; (c) inverse model, $A = I(O, E)$. With respect to planning, these functions can play various roles. "Forward models" can be used to support forward planning, as here, because they allow the agent to predict the effect of performing a given action A on a certain object O and check if the obtained effect E matches a desired goal/sub-goal (G). "Relevance" checking (Russell and Norvig, 2016) allows the agent to search actions for which the effect fulfills the goal and then to search for relevant objects (on which the actions can be applied) to accomplish the goal (the function C has also been used often in the affordance literature to perform sheer object recognition based on how objects respond to actions, e.g., see Fitzpatrick et al., 2003; Castellini et al., 2011; Nguyen et al., 2013; Ugur and Piater, 2014). "Inverse models" can be used to directly perform actions, with the caveat that the object does not represent the whole state of the environment as required by proper inverse models (this is not further discussed here). These broad definitions of affordance are important to highlight the triadic relational nature of affordances, involving not only *object/conditions* and *actions* but also *action effects*. On the other hand, such definitions overlap to a certain degree with other concepts used in the computational literature and this decreases their utility.

Here we contribute to the investigation of the possible functions of affordances for autonomous agents by assuming a restricted definition of them. This definition allows us to evaluate the utility of affordances within planning systems and also to contribute to clarifying the relation between the concept of affordance used in psychology and in robotics. Informally, the definition is this: *an affordance is an agent's estimated probability that a certain action performed on a certain object successfully accomplishes the desired outcome associated with the action ("goal").* Formally, the definition of affordance used in this work is as follows:

> *An affordance is an agent's estimated probability $Pr(s'_{b,o} \in G|a, s_{b,o})$ that if it performs action a on the object o when the object and own body b are in state $s_{b,o}$ then the outcome will be a state $s'_{b,o}$ belonging to a set of goal states G to which the action is directed.*

We illustrate the features of this definition and its differences and links with other definitions. (a) The definition is more specific than other definitions that often have vague features. (b) The states $s_{b,o}$ refer to a certain object and the agent's body so the definition is closely linked to the original idea of affordance as founded on the body-object relation. Moreover, the focus on "b" and "o" differentiates the concept from the *transition models* (linking current state and action to future states) used in the reinforcement learning literature (Sutton and Barto, 2018) as these use *atomic/whole-state representations*; instead, such reference links the definition to the *preconditions* used in symbolic planning operators that use *structured representations* capturing the relations between different entities (here "o" and "b"; Russell and Norvig, 2016). (c) The definition is centered on the concept of "object," intended here as a limited portion of matter that is physically detached from the rest to the environment. This is very important as in the literature on affordances objects are crucial for *manipulation tasks*, an important class of robotic tasks alongside *navigation tasks.* As discussed below, the focus on objects makes attentional processes very useful for the acquisition and use of affordances. (d) Importantly, the definition is grounded on the link between the action and its *goal*, i.e., the possible effects that the action is expected to produce on the object; for example, a "grasping action" might have the goal "hand envelops the object." This is important as often in the computational literature on affordances it is assumed that the agent is able to check the success of an action performed on a certain object (e.g., that "the object rolls" if pushed) without fully recognizing that such a check needs a reference state/event with which to compare the action outcome; that is, it needs a goal. (e) The goal can be abstract in the sense that it encompasses different states $s_{b,o}$. In particular, it can be abstract with respect to the elements of the environment other than the object and body. Sometimes the goal might even be abstract with respect to the body features; for example different robots with different actuators could all be able to "grasp" a certain object. In some cases, the goal might abstract altogether from the body and involve more than one object, in particular some relation between them; for example the goal might require piling "object A on object B" (in this case the affordance is that A is "pileable" on B). The "o" in the definition might even go beyond "objects" and include broader affordances; for example Ugur et al. (2007) studied the "traversability" of a portion of environment in a navigation task. (f) There can be many different possible goals G, and actions to pursue them, that an agent can perform on a given object; for example a robot can "push," "grasp," or "lift" a given object to accomplish different goals. (g) The definition, based on a probability, takes into account the uncertain nature of the environment where the performance of actions does not necessarily produce the desired outcomes. (h) The definition assumes a binary success/failure of the action, for example based on the use of a threshold (e.g.: "the object is considered as reached if the distance between the object and the hand-palm after action execution is smaller than 2 cm"). An alternative definition, assuming that suitable distance metrics could be applied to the object/body states, could state that the affordance is accomplished *in a continuous degree* related to the final distance of the state achieved by the action and the reference goal-state (e.g.: "a reaching action toward an object brought the hand 5 cm close to it"). Here we will use the first definition related to probability-based binary affordances. Within this, we will consider two types of affordances. The first, called here *deterministic affordances*, where objects *allow or not* the agent to accomplish the goal (e.g., an object could be "movable" or "non-movable"): in this case the affordance probability is equal to 0 or 1. The second, called here *stochastic affordances*, can accomplish the goal only with a certain probability, for example an object might be "movable" with a probability of 0.7 and another one with a probability of 0.3.

We now introduce a pivotal feature of our system, the use of *active vision* (Ballard, 1991; Ognibene and Baldassare, 2015). Agents with active vision: (a) use a visual sensor that returns information related to only a limited portion of the scene; (b) actively direct the sensor on relevant regions of space ("overt attention"). These assumptions reflect fundamental principles of organization of the visual system of primates (Ungerleider and Haxby, 1994) and in artificial systems they allow the reduction of visual information processing and an easier analysis of spatial relations between scene elements (Ognibene and Baldassare, 2015). Here active vision is used to extract information on objects, very important for three processes: intrinsic motivations, affordance processing, and planning. Regarding intrinsic motivations, we shall see that our system relates them to objects and on this basis decides on which object to invest exploration and learning. Regarding affordances, we have seen above that most works in the literature use setups structured so that the system can gather information on affordances related to *single objects* (e.g., see Fitzpatrick et al., 2003). The system proposed here uses active vision to focus on single objects and detect their affordances. Regarding planning, the classic AI planners usually employ *structured knowledge representations* of states and actions ("operators") expressed with propositional logic or first order logic; this type of representation is very important as it allows systems to "reason" about objects and their relations, and "almost everything that humans express in natural language concerns objects and their relationships" (Russell and Norvig, 2016). For our focus on embodied systems, we instead use here *factored knowledge representations* based on feature vectors, in particular image features. However, active vision allows the system to identify single objects, and this information allows the system to parse overall goal images into object-related sub-goals that can be pursued one by one. The planning processes considered here are akin to those used within the *Dyna systems* of reinforcement learning literature where planning is implemented as a reinforcement learning process running within a world model rather in the actual environment (Sutton, 1990; Baldassarre, 2002; Sutton and Barto, 2018). Some caveats on the approach used here to visually isolate objects are due. We simplify the task by not considering cluttered scenarios. Moreover, we use simple bottom-up attention processes to control gaze. These simplifications allow us to develop the overall architecture of the system, but in future work some of the components of the system could be substituted by more sophisticated components, in particular for object segmentation and detection (Yilmaz et al., 2006; Zhang et al., 2008; LeCun et al., 2015) and for a smarter "top-down" control of gaze depending on the agent's information needs (Ognibene et al., 2008, 2010; Dauce, 2018).

1.2. Contributions of the Work

A first contribution of this study is on how intrinsic motivations can support efficient learning of affordances, in particular when an attention mechanism focusing on objects is used. There are some previous studies linking affordance learning to intrinsic motivations and active learning (Ugur et al., 2007; Nguyen et al., 2013), but they did not investigate how intrinsic motivations can be used to learn object affordances when the visual sensors access only one object at a time. To face this condition, we will propose a mechanism that compares the estimated learning progress from acting on the currently seen object with the progress that it could gather by acting on other objects. The mechanism is inspired by the concept of *opportunity cost* used in economics, referring to the value of the opportunities that are lost by allocating a certain resource (here a unit of learning time) to a certain activity (Buchanan, 2008).

A second contribution of this work is the study of how the introduction of the attention mechanism, extracting information about the single object and about the object appearance/location impacts (a) the affordance learning process and (b) the second extrinsic phase where planning is needed to accomplish an extrinsic complex goal. The first issue has only been indirectly studied in the literature on affordances where models often assume pre-processing mechanisms to extract information on specific objects (see also the "OAC – Object Action Compound" framework pivoting on object information; Krüger et al., 2011). The second issue is important as attention is a key means to detect objects in humanoid robots (Camoriano et al., 2017) and information on objects is pivotal for both affordance detection (e.g., Montesano et al., 2008) and for planning. Regarding planning, attention can be used to refer to single objects, and then to reason about their relations, as typically done when using *structured representations* (Russell and Norvig, 2016). In particular, in the intrinsic phase the architecture presented here can use attention to identify different objects in the environment and learn affordances related to them. Then in the extrinsic phase it can use attention to parse the whole goal state into object-related sub-goals that can then be more easily accomplished one by one.

A third contribution of this work concerns the relationship between affordances and planning. In particular we will face the problem of what could be the utility of affordances, defined as the probability estimate of action success, within a planning system that is endowed with refined components implementing *forward models* and *relevance checking*. Are affordances still useful in such conditions? The importance of this problem derives from the fact that psychology shows that affordances are very important in real organisms. One might thus wonder if they can still furnish relevant functions within sophisticated systems endowed with the capacity of planning. In this respect, we will propose that: (a) affordances can play a role in forward planning as they support fast selection of relevant actions within the system's controller in a way similar to the way they are used to act in the environment, akin to the role of the "preconditions" of STRIPS-like operators in symbolic planning (Fikes and Nilsson, 1972); (b) affordances, when capturing the expected probability of action success, can play an important role in utility-based planning agents, the most sophisticated form of rational agent (Russell and Norvig, 2016).

The rest of the paper is organized as follows. Section 2 illustrates the experimental setup, and the architecture and functioning of the system. Section 3 shows the results of the tests. Section 4 compares the system proposed here to other systems proposed in the literature. Finally, section 5 draws the conclusions and illustrates open problems that might be tackled in the future.

2. METHODS

2.1. Experimental Setup: Overview

The experimental scenario (**Figure 1**) consists of a black 2D working space containing different objects. Objects have different shapes (squares, circles, and rectangles) and colors (red, green, and blue). Section 2.2 describes the scenario and objects in more detail.

Each test consists of two phases: the *intrinsic phase* and the *extrinsic phase*. In the intrinsic phase the agent is free to interact with the objects for a certain time to autonomously acquire knowledge on them, in particular on their affordances and on forward models related to them (**Figure 1A**). In the extrinsic phase an overall goal state, with specific desired location and color of the objects in the working space, is presented to the agent that memorizes an image of it. The objects are then set in a different state and the agent has to re-create the goal state based on knowledge acquired in the first phase (**Figure 1B**).

The agent (section 2.3) is endowed with a simulated camera sensor that can look at different sub-portions of the working space, and is able to select and perform four actions on the object that is at the center of its camera. The actions can move the object to a new position or change its texture to a particular color (red/green/blue); different objects afford only a subset of these actions. As often assumed in the affordance literature, action execution is based on pre-programmed routines implementing the effect of the action that is selected and triggered.

Three versions of the system are implemented and compared: IGN, FIX and IMP. The three systems differ in the IM mechanisms they use to support affordance learning: FIX uses a mechanism taken from the literature (Ugur et al., 2007) whereas IGN and IMP use new mechanisms. Section 2.4 describes the three systems in detail. In the extrinsic phase, the knowledge acquired by the three systems in the intrinsic phase is tested with problems requiring goal-based or utility-based planning.

2.2. Working Space and Objects

The working space is formed by a 150 $\times$ 150 pixel square. The points of the working space are encoded as a 3D binary array where the first 2 dimensions encode the x-y pixel position, and the third dimension encodes the color (RGB). The color of the working space background is black. All objects are initially located on the vertexes of a 3 $\times$ 3 regular grid (white square in **Figure 1**).

Each object presents the following attributes: (a) center: x-y coordinates; (b) color: three values for red, green, and blue; (c) shape: either square, circle, or rectangle. Assuming the working space has a side measuring 1 unit, the circle has a diameter measuring 0.1 units, the square has a side measuring 0.1 units, and the rectangle has sides measuring 0.6 and 0.16 units.

As a consequence of the actions performed by the agent, the position and color of the objects can change. This defines the possible affordances of objects: "movable," "greenable," "redable," and "bluable." Each object has a specific subset of affordances, for example a blue circle is "movable" and "redable." In some tests, affordances are stochastic in the sense that the related actions can produce an effect only with a certain probability.

2.3. The System Architecture

The system controller consists of three different components (**Figure 3**): (a) the *perception component* implements a "bottom-up attention" mechanism that leads the system to scan the environment based on its salient features (color blobs of objects) and also observes the action effects in the environment by looking at portions of the scene that are changed by the actions (section 2.3.1); (b) the *action component* executes actions on the objects based on pre-wired routines (section 2.3.2); (c) the *predictor component* is formed by the predictors for affordances and forward models that support action selection in both the intrinsic and extrinsic phases (section 2.3.3). In the following sections we describe the different components in more detail.

2.3.1. The Perception Component: Attention and Effect Detection

The perception component is responsible for the attention processes supporting visual exploration of both the environment and, in the extrinsic phase, the goal image. The perception component implements two attention processes: an inner attention process operating in parallel with an outer attention process. The outer attention scans the environment on the basis of two bottom-up processes both affecting gaze (they sum up): the first process is sensitive to the saliency of objects, and the second process is sensitive to the changes of the appearance of objects produced by actions. The inner attention scans the goal image on the basis of either the saliency of objects or by having the same focus as the one of the outer attention process. All these

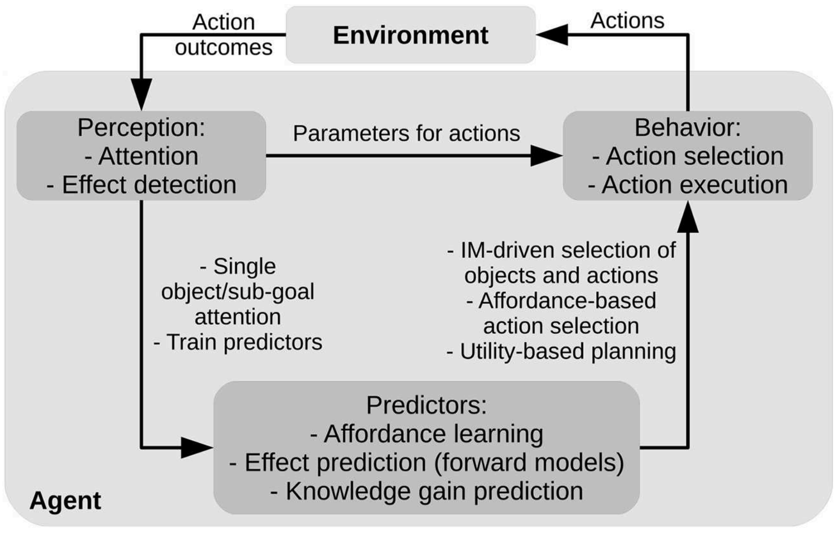


FIGURE 3 | System architecture: main components.

attentional processes are activated when needed on the basis of intrinsic motivations or planning processes, as we now illustrate more in detail.

Attention actively guides a RGB visual sensor (a pan-tilt camera) returning an image centered on the current attention focus and sufficient to always cover the whole working space independently of the gaze pointing. The central part of such an image, called *focus image*, forms the main input of the system. This focus image has a size of 0.14×0.14 (recall the scene is 1×1) and usually covers only one specific object. The whole image is used with a lower resolution (here in gray scale) to form a *peripheral image* that is used to drive the bottom-up attention processes now illustrated (cf. Ognibene and Baldassare, 2015).

The first type of bottom-up attention process, the saliency-based one, is driven by the most "salient" elements in the peripheral image, here the activation of pixels corresponding to objects. This process is implemented as follows. First a random noise ($\epsilon \in [-0.05, 0.05]$) is added to each pixel of the peripheral image and the resulting image is smoothed with a Gaussian filter. Then the pixel with the maximum activation is used as the focus of attention (but if a change happens the second bottom-up attention mechanism also intervenes, see below). Thanks to the Gaussian smoothing, the focus falls around the center of the focused object, which thus becomes wholly covered by the focus image. The noise fosters exploration as it adds randomness to the saliency of objects, thus leading the system to explore the different objects.

Note that in the future more sophisticated approaches might be used to ensure that the focus-image involves only the focused object of interest (e.g., object-background segmentation approaches). Moreover, other mechanisms might be used to ensure a more efficient scan of the environment (e.g., inhibition of return might be used to avoid scanning the same location multiple times). These mechanisms are not considered here for simplicity and because the focus of this research is on the *effects* of attention, rather than on the mechanisms for its *control*.

The second bottom-up attention process, sensitive to changes, is directed to detect the effects of actions. The process works as follows. Firstly the system focuses on the portion of space where a *change* in the periphery image takes place (this mimics some processes of primates for which a reflex focuses attention on changes happening in the environment (Comoli et al., 2003; Gandhi and Katnani, 2011; Sperati and Baldassarre, 2014). To this purpose, the system computes the "change image" given by the pixel-by-pixel absolute difference between the whole periphery image after and before the performance of the action: both images are taken with the same initial gaze before the next attentional movement but the "after-image" is taken after the action performance (e.g., leading to displace the object). The change image is smoothed with a Gaussian filter and summed up to the salience-based attention image described above to guide the gaze displacement to the maximally activated pixel of the "sum image." This process leads the system to look at the object that has been changed by the action, e.g., the object that has been changed in color or displaced in space (in the latter case, the system gazes the position where the object is moved, and not the position where it disappears, due to the object saliency that is not present where it disappears). After this attentional movement on the changed object is performed, the system compares the focus image (involving only the area covering the object) and the object position (given by the gaze direction) before and after performing the attentional movement itself: this allows the system to decide if the performed action was successful or not (presence of affordance). The focus image comparison is based on the L1 norm of the difference between the vectors of the two images before and after action performance, divided by the number of dimensions of the vectors: if this measure is higher than $\tau_i = 10^{-5}$ then the focus image is considered to have changed. The position comparison is based on the L1 norm of the difference between the vectors of the two (x,y) focus positions before and after action performance, divided by 2: if this measure is higher than $\tau_p = 0.02$ then the position of the object is considered to have changed.

An affordance is considered to be in place if: (a) an effect is detected; (b) the effect is the one related to the performed action (e.g., the object is displaced by the action "move," or the object is made green by the action "change object color to green"). This information is used to train the affordance predictors. The focus image and object position after the action execution are used to train the effect predictors.

2.3.2. The Action Component

The system is equipped with four actions: move object, change object color to green, change object color to red, and change object color to blue. The move action can displace objects in the environment if they have the affordance for this effect. The move action is parametric: it affects the target object (object under focus) on the basis of two parameters corresponding to the object desired x-y location. During the intrinsic phase, the desired location is randomly generated within the working space (excluding positions that cause object overlapping). During the extrinsic phase, the target location corresponds to the location of the "sub-goal" that the system is currently attempting to accomplish (see section 2.4.3). The color-changing actions are non-parametric: they simply change the color of the target object into the desired one if the object has the affordance for the corresponding effect. Only one color-change action might have been considered if parameterized with the color (this would have been a discrete parameter vs. the continuous parameters of the move action). We chose a non-parametric version of the color actions to develop the features of the system working for both parametric and non-parametric actions.

2.3.3. The Predictor Component: Forward Models and Affordances

The predictor component is formed by 16 predictors (these are regressors), 4 for each of the 4 actions: (a) *the affordance predictor* predicts the object affordance (i.e., the probability that the action effect takes place when the action is performed); (b) *the learning-progress predictor* predicts the learning progress of the affordance predictor when applying the action to the target object, and is used to generate intrinsic motivations based on the learning progress of the affordance predictors; (c) *the what-effect predictor* predicts the focus image of the object resulting after the action performance; (d) *the where-effect predictor* predicts the object position resulting after the action performance. Given the simplicity of the stimuli, the predictors are implemented here as simple perceptrons but more sophisticated models might be used to face more challenging scenarios. All the predictors are trained during the intrinsic exploration phase and are now explained more in detail.

The affordance predictors estimate the affordance probability $Pr(s'_{b,o} \in G|a, s_{b,o})$ of each action/goal related to different objects. Each predictor gets as input the focus image (whose pixels are each mapped onto $(0, 1)$ and unrolled into a vector) and returns, with one output sigmoid unit, the prediction of the action success. Each predictor is trained with a standard rule and a learning target 0 or 1, encoding respectively the failure or success of the action to produce its desired effect, i.e., the presence/absence of the affordance (the learning rate used varied in the different tests, see section 3).

Each learning-progress predictor gets as input the focus image and returns, with a continuous linear output unit, the learning progress of the associated affordance predictor. The predictor is updated with where the target for learning is the difference in the output of the corresponding affordance predictor, computed before and after the action is performed and before the affordance predictor is updated.

Each of the what-effect predictors gets as input the focus image and predicts, with sigmoidal output units, the focus image after the action performance. The predictor is updated with rule with a target corresponding to the observed focus image after the action is performed.

Each of the where-effect predictors gets as input the initial (x, y) position of the target object and the desired (x, y) position of the object depending on the sub-goal, and predicts, with two linear units, the predicted object (x, y) position after the action performance [x and y coordinates are each mapped to the range $(0, 1)$]. The predictor is updated with a where the target for learning is the object position after the action is performed.

2.4. The Intrinsic-Phase Learning Processes and the Extrinsic-Phase Planning

In this section we first present the motivation signals (section 2.4.1) and the algorithm for learning affordances and forward models (section 2.4.2) used by the three versions of the system (IGN, FIX, and IMP) during the intrinsic phase. Then we describe the two algorithms of the attention-based goal planner (section 2.4.3) and the attention-based utility planner (section 2.4.4) used in the extrinsic phase.

2.4.1. IM Signals

In the intrinsic phase, the system autonomously explores the objects in the environment to learn affordances and train its predictors. The exploration process is driven by IMs related to the knowledge acquired by the affordance and learning-progress predictors. Depending on how the IMs are implemented, we have three versions of the system: FIX, IGN, and IMP.

The FIX system uses an IM mechanism for affordance learning like the one used by Ugur et al. (2007). This work studies a mobile robot that learns the "traversability" affordance in a maze scattered with obstacle-objects. A Support Vector Machine (SVM) is used to classify the view of the obstacle-objects to estimate the presence/absence of the affordance. The system estimates the novelty, and hence the interest, of the current view of the objects on the basis of its distance from the hyperplane used by the SVM to classify the affordance presence/absence. If this distance is below a fixed threshold, the view is considered interesting and so the system performs the exploratory action of trying to traverse the maze. The system observes if the affordance holds, and uses its observation to train the SVM.

In our case, as a measure of how interesting the current object is we consider the Shannon entropy of the estimated affordance probability:

$$H(p) = -\sum_{i=1}^{n} Pr(x_i) log_b(Pr(x_i)) = -p\log_2 p - \left(1-p\right)\log_2\left(1-p\right) \tag{1}$$

where we considered $b = 2$ as the basis of the logarithm; x_i are the two events $s'_{b,o} \in G$ and $s'_{b,o} \notin G$ corresponding to the presence or absence of the affordance, having respectively a probability $Pr(s'_{b,o} \in G) = p$ and $Pr(s'_{b,o} \notin G) = 1 - p$. The use of this formula is justified by the fact that entropy is a measure of ignorance (uncertainty) of the system: the uncertainty is minimal when $p = 0$ or $p = 1$, and maximal when $p = 0.5$ (the value of the entropy is here normalized so that $H(p) \in (0, 1)$, in particular $H(0) = H(1) = 0$ and $H(0.5) = 1$). The weights of the affordance predictors are initialized to 0, resulting in an initial sigmoid activation of 0.5 and an ignorance of 1.

Following Ugur et al. (2007), the current object is considered interesting, and hence worth exploring, when the entropy is above a threshold th (here $th = 0.3$, which corresponds to an ignorance value, i.e., affordance predictor output, of 0.947 or 0.053). Ignorance (entropy) thus represents the IM signals that drive exploration of objects. Since we have more than one action, for a given focused object we consider the action with maximum ignorance for that object. This mechanism is simple and interesting, but it also has some limitations when applied to multiple objects and actions as it leads to an evaluation of how interesting potential experiences are in a fixed way. In particular, all objects with an ignorance above the threshold will be considered equally interesting. Moreover, after the ignorance related to an object decreases below the threshold the agent will stop exploring it independently of the fact that it might still have some exploration time available.

The IGN system (IGN stands for "IGNorance") is a first version of our system that is directed to overcome the limitations of the IM mechanism of FIX. The new mechanism uses a dynamic threshold th. This threshold is continuously adjusted as a leaky average of the IM signal, IM (here $IM = H$), related to the objects explored one after the other in time:

$$th_t = th_{t-1} + \nu(-th_{t-1} + IM_{t-1}) \tag{2}$$

where t is a trial and ν the leak coefficient ($\nu = 0.3$ in the deterministic environment and $\nu = 0.1$ in the stochastic environment). When objects are explored and predictors are trained, the ignorance of objects, and hence the IM signal related to them, will decrease and so will th. By comparing the highest action-related entropy of an object with th the system will be able to trigger the exploration action or to pass to explore more interesting objects. When the system decides to not explore the object, then the leaky average gets $IM_{t-1} = 0$ as input. This implies that when objects are considered not interesting, the threshold progressively decreases so that some objects become interesting again.

A limitation of the IM mechanisms of IGN is that it is not able to cope with stochastic environments where the success of an action is uncertain (e.g., when the agent tries to move an object, the object moves only with a certain probability). In this case, the affordance predictor will tend to converge toward the corresponding probability p and so objects with stochastic affordances will always remain interesting.

The IMP system (IMP stands for "IMProvement") overcomes the limitations of IGN by suitably coping with stochastic environments. The motivation signal used by IMP is implemented as the absolute value of the learning-progress predictor output (let's call this LP). Similarly to IGN, th is computed as a leaky average of the IM signal, namely with Equation (2) where $IM = LP$ ($\nu = 0.1$ in all tests). In particular, as for IGN the expected learning progress of the action with the highest LP is compared with th. This makes the system explore the current object if it promises a learning progress that is higher than the learning progress for other objects, represented by th. The learning-progress predictor weights are initialized to random values within $(-0.00075, +0.00075)$ so that the motivation signal is non-zero for all objects when the intrinsic phase starts. The distinction between IGN and IMP reflects the distinction between *prediction error* and *prediction error improvement* within the literature on IMs, justified as here by the need to face deterministic or stochastic environments (Schmidhuber, 1991a; Santucci et al., 2013).

The mechanism of the leaky average threshold, used in IGN and IMP, allows the agent to indirectly compare the relative levels of how interesting different objects are, and to focus the exploration effort on the most interesting of them notwithstanding the fact that *different objects are in focus at different times* due to the presence of the active vision mechanisms. This is a novel general mechanism that allows the integration of attention (targeting different objects) and IMs (returning an interest of the object). Here the mechanism is used for the learning of object affordances but its generality allows its use also for the acquisition of other types of knowledge in the presence of focused attention.

2.4.2. Intrinsic Phase: Learning of Affordances and World Model

The intrinsic phase allows the system to autonomously explore the objects by looking at and acting upon them. Based on the observed consequences of actions, this allows the system to train the predictors, including those related to affordances. At each step of the phase, the system performs a number of operations as illustrated in **Algorithm 1**. This algorithm is used by both the goal-based planner and the utility-based planner.

The algorithm is based on the following operations and functions: (a) *the Scan* function focuses the system visual sensor on an object based on the bottom-up attention mechanism and returns the image and position of the object; (b) *SelecActionWithHighestIM* selects the action with the highest IM and computes its motivation signal; (c) *ExecuteAction* executes the selected action as illustrated in section 2.3.2 if the motivation signal is higher than the threshold; (d) *ScanEffect* looks where a change in the environment has happened (e.g., at the new position occupied by the object after the move action, or at

Algorithm 1: Intrinsic phase: one step of learning of affordances and forward models

```
1:  (object_image, object_position) ← Scan(environment)
2:  (action, motivation_signal) ← SelectActionWithHighestIM(action_list, predictors, object_image, object_position)
3:  if (motivation_signal ≥ motivation_threshold) then
4:      ExecuteAction(action, object_image, object_position)
5:      (new_object_image, new_object_position) ← ScanEffect(new_environment, environment)
6:      affordance ←...
7:          Affordance(action, new_object_image, new_object_position, object_image, object_position)
8:      UpdateWeights(affordance_predictor, action, object_image, affordance)
9:      UpdateWeights(affordance_predictor, action, object_image, affordance, improve_predictor)      ▷ Only IMP
10:     if (affordance = TRUE) then
11:         UpdateWeights(effect_predictors, action, object_image, object_position,...
12:             new_object_image, new_object_position)
13:     end if
14:     motivation_threshold ← LeakyAverage(motivation_threshold, motivation_signal)      ▷ Only IGN/IMP
15: else
16:     motivation_threshold ← LeakyAverage(motivation_threshold, 0)      ▷ Only IGN/IMP
17: end if
```

the object that changed its color after a change-color action), and returns the resulting new object image and position; (e) *Affordance* compares the past and new state of the target object and returns a Boolean value (affordance) for the affordance presence/absence; (f) *UpdateWeights* updates the connection weights of the predictors of the performed action as illustrated in section 2.3.3: the affordance predictor is updated on the basis of the affordance presence/absence; in the case of IMP, the learning-progress predictor is also updated on the basis of the affordance predictor output before and after the action performance (and hence the predictor update); in the case of the presence of the affordance (action success), the effect predictor learns to predict the effects of the performed action; (g) *LeakyAverage*, present in the case of IGN/IMP, updates the threshold according to Equation (2) and uses as input either the IM signal of the performed action, or zero if no action was executed.

2.4.3. Extrinsic Phase, Pursuing the Overall Goal: Goal-Based Planner

During the extrinsic phase, the system is tested for its capacity to accomplish an "overall goal" based on the knowledge acquired during the intrinsic phase. Such an overall goal is assigned to the agent through the presentation of a certain desirable spatial/color configuration of some objects in the environment. The agent stores the goal as an image ("goal image"). The configuration of the objects is then changed and the task of the agent is to act on the environment to arrange it according to the goal image.

Importantly, the agent scans the goal image through a second "inner" attention mechanism similar to the "outer" attention mechanism used to scan the external environment. This inner attention mechanism is important to parse the goal image into *sub-goals*, each corresponding to the configuration of a single object in the goal image, one per saccade. These sub-goals are then pursued in sequence to accomplish the overall goal.

The operations taking place in one step of this process are shown in detail in **Algorithm 2**. The pseudo-code of the algorithm highlights the effects of the factorization of information on objects, related to their state and position, based on the attention mechanisms. A new sub-goal is selected either in the case that the previous sub-goal has been accomplished (in which case the Boolean variable *sub_goal_active* $=$ *FALSE*) or if a time out elapses in the unsuccessful attempt to pursue it (here the time out is equal to 8 iterations of the algorithm). The agent uses the function *Scan* to identify a new target sub-goal: this function scans the goal-image with the saliency-based attention mechanism and returns the new sub-goal image and focus location (*sub_goal_image* and *sub_goal_position*).

Next, the agent checks if the sub-goal has not been accomplished yet. To this purpose, the function *ScanEnvironmentWithSameFocusAsSubGoal* drives the outer attention focus (targeting the environment) to the position corresponding to the inner attention focus (targeting the goal image) and returns the corresponding focus image (*focus_image*). Then the function *GoalNotAchievedCheck* compares the *sub_goal_image* and the *focus_image* to check that the sub-goal has not been achieved yet, in particular it sets the variable *sub_goal_active* to *FALSE* or *TRUE* if they, respectively, match or mismatch (the match holds if the Euclidean distance between the vectors corresponding to the two images is below a threshold $\tau = 0.01$).

If the sub-goal has not yet been accomplished, the system scans the environment to find a new object and then the function *ActionPlanning* checks if at least one action is able to accomplish the sub-goal by acting on the focused object. To this purpose, the function uses the effect predictors to predict the effect of each action and then compares it with the sub-goal (this happens if the Euclidean distances are below 0.0035 for the sub-goal image and the object image, and below 0.01 for their position coordinates). Depending on the result of the planning, the function sets the Boolean variable *sub_goal_achievable* to *TRUE* or *FALSE* and possibly returns the action to be executed to achieve the sub-goal. These processes, based on the forward models, are an important

Algorithm 2: Extrinsic phase: one step of goal-based planning

```
if time_out OR (NOT sub_goal_active) then                                    ▷ Select non-achieved sub-goal
    (sub_goal_image, sub_goal_position) ← Scan(goal)
    focus_image ← ScanEnvironmentWithSameFocusAsSubGoal(environment)
    sub_goal_active ← GoalNotAchievedCheck(sub_goal_image, focus_image)
end if
if (sub_goal_active = TRUE) then
    (object_image, object_position) ← Scan(environment)                                  ▷ Select object
    (sub_goal_achievable, action) ← ActionPlanning(predictors, action_list,...
        sub_goal_image, sub_goal_position, object_image, object_position)                  ▷ Plan action
    if (sub_goal_achievable = TRUE) then
        ExecuteAction(action, object_image, object_position, action)                    ▷ Perform action
        focus_image ← ScanEnvironmentWithSameFocusAsSubGoal(environment)
        sub_goal_active ← GoalNotAchievedCheck(sub_goal_image, focus_image)
    end if
end if
```

part of the algorithm as they implement a one-step forward planning process.

Lastly, if a potentially successful action has been identified, it is executed and then the system checks again if the sub-goal has been accomplished. If so, *sub_goal_active* is set to *FALSE* so that a new sub-goal is chosen in the next iteration.

Importantly, the *ActionPlanning* function could possibly check all actions. Affordances (i.e., the predictors estimating $Pr(g_j|a_j, o_i)$) can be employed to avoid this. In particular, the check can be limited only to those actions having an affordance for the current object (here when $Pr(g_j|a_j, o_i) > 0.5$). This allows affordances, which are computed fast through 1-output neural networks, to speed up the planning search by reducing the number of more computationally expensive operations involving the prediction of action effects (new object image and position) and their comparison with the sub-goal. Here actions are only 4, but this advantage increases with the number of actions available to the agent.

2.4.4. Extrinsic Phase, Pursuing the Overall Goal: Utility-Based Planner

In the case of *utility planning*, different "sub-goals" deliver a different value if accomplished. In cases where the agent has a limited amount of resources available to accomplish the goals (e.g., time or energy to perform actions), it should first invest such resources in the accomplishment of the most valuable sub-goals (for simplicity, we assume here a constant cost per action and a negligible cost of reasoning with respect to acting, as often done in utility-based planning, Russell and Norvig, 2016). Notice that in utility planning "sub-goals" are directed to accomplish the overall objective of utility maximization rather than an overall goal intended as a particular state of the environment.

The utility-based planner works as shown in **Algorithm 3**. When the Boolean variable *max_utility_estimatation* is *TRUE*, the planner evaluates the value of the possible sub-goals it can achieve with the available object-action combinations and stores an estimate of its value in the variable *potential_utility*, otherwise it acts in the world. Various mechanisms could be used to set and keep the system in the evaluation mode: here for simplicity we gave the system a certain amount of iterations before performing an action, but more flexible mechanisms might be used (e.g., passing to act when the estimates stabilize). To perform this evaluation, the system performs the sub-goal seeking, object seeking, and one-action planning processes as done in **Algorithm 2**. However, instead of executing the planned actions the system only updates the *potential_utility* if the current goal-object couple has a higher utility than it: this ensures that the potential utility estimation tends to approximate the value of the most valuable sub-goals. The utility of the sub-goal-object couple, given the found action, is computed as:

$$U = Pr(s'_{b,o} \in G|a, s_{b,o}) \times V(s'_{b,o}) \tag{3}$$

where $Pr(s'_{b,o} \in G|a, s_{b,o})$ is the affordance (expected probability of accomplishing the desired sub-goal) and $V(s'_{b,o})$ is the value of the sub-goal. The actual update of the potential utility is based on a leaky average (based on Equation 2 using a leak rate $\nu = 1.0$; a value lower than this might be used for having a more reliable but slower process).

When the *max_utility_estimation* is set to *FALSE*, the system starts to perform actions in the environment. This is similar to what is done in **Algorithm 2**, the difference being that now the system executes actions only if the expected utility of the current subgoal-object is higher than the potential utility. For each chosen action, this potential utility is decreased with the leaky average using 0 as input and a leak rate $\nu = 0.1$. This ensures that the potential utility progressively decreases so that the system initially works on most valuable sub-goals-object couples and then engages with less valuable ones.

3. RESULTS

To test the performance of the systems, different tests were run with both deterministic and stochastic environments. Performance in the intrinsic phase was measured by evaluating the quality of the output of the predictors when receiving as input each one of the nine focused images corresponding to the nine possible objects (**Figure 1**).

Algorithm 3: Extrinsic phase: one step of utility-based planning

```
if (time_out OR (NOT sub_goal_active)) then                                   ▷ Select non-achieved sub-goal
    (sub_goal_image, sub_goal_position) ← Scan(goal)
    focus_image ← ScanEnvironmentWithSameFocusAsSubGoal(environment)
    sub_goal_active ← GoalNotAchievedCheck(sub_goal_image, focus_image)
end if
if (sub_goal_active = TRUE) then
    (object_image, object_position) ← Scan(environment)                                     ▷ Select object
    (sub_goal_achievable, action) ← ActionPlanning(predictors, action_list,...
        sub_goal_image, sub_goal_position, object_image, object_position)                    ▷ Plan action
    if (sub_goal_achievable = TRUE) then
        object_utility ← ComputeUtility(object_affordance, sub_goal_value)
        if (max_utility_estimatation = TRUE) then                ▷ Computing the maximum possible utility
            if (object_utility ≥ potential_utility) then
                potential_utility ← LeakyAverage(potential_utility, object_utility)    ▷ Increase utility expectation
            end if
        else                                                               ▷ Acting if high utility is attainable
            if (object_utility ≥ potential_utility) then
                ExecuteAction(object_image, object_position, action)
            end if
            potential_utility ← LeakyAverage(potential_utility, 0)            ▷ Decrease utility expectation
        end if
        sub_goal_active = FALSE
    end if
end if
```

3.1. Deterministic Environment

In the deterministic environment two tests were run to test the goal-based planner. The first, called the *base test*, involved all nine objects each affording all four actions. The purpose of this test is to compare the different intrinsic motivation mechanisms driving affordance acquisition. During the intrinsic phase of this test, the objects are initialized as in **Figure 1A** and the system explores them. Afterwards, in the extrinsic phase the system is tested in five different conditions involving different goal images and environment settings (**Figure 4**). The first scenario contains only one object, a blue square that has to be changed to green. In each subsequent scenario one additional object is introduced to increase the scenario difficulty: scenario 2 introduces a green rectangle that has to be changed to red; scenario 3 introduces a red square that has to be moved; scenario 4 introduces a green circle that has to be changed to blue; finally, scenario 5 introduces a red circle that has to be moved.

The second test, called the *late object test*, involves an intrinsic phase where some objects, not initially present, are introduced after the system has acquired knowledge on objects introduced initially. Then the system is tested with the extrinsic phase scenarios as in the base test (**Figure 4**). The purpose of this test is to evaluate how the systems perform when, during the intrinsic phase, new knowledge is added to already acquired knowledge, a situation very common in open-ended learning conditions.

Affordance predictor learning rates were $\alpha = 0.01$ in the IGN and IMP systems and $\alpha = 0.002$ in the FIX system. The learning rates of the learning-progress predictors were set to $\alpha = 0.005$. The leaky average of the intrinsic motivation was updated with a leak rate $\nu = 0.3$ in the IGN system and $\nu = 0.1$ in the IMP system. The results of these tests are presented in the following sections.

3.1.1. Base Test

In the base test, all the nine objects afford all the four actions with the exception of those not causing any change (**Table 1**).

Intrinsic phase. The intrinsic phase of the base test was run 10 times, each lasting 6,000 steps, for each system, IGN, FIX and IMP. All three systems learned a good estimate of the affordances of the nine objects (**Figure 5**; the true values to estimate are either 0 or 1). The affordance predictions after learning were closer to the true values for the IGN and IMP systems compared to the FIX system. The variance of the predictions was lower in the FIX and IGN systems compared to the IMP system **Figure 5A**.

These results offer a first validation of the idea that the IMP and ING systems, using a dynamic threshold for evaluating the interest of the current object in terms of its potential return of information, outperform the FIX system previously proposed in the literature. The reason is that the IMP and IGN systems can decide to explore or ignore an object based on the possibility of learning more from other objects, rather than in absolute terms as in FIX.

Extrinsic phase. The test of the extrinsic phase was repeated 10 times for each system, for each extrinsic scenario, and for each of the 10 repetitions of the intrinsic phase. The results show that the three systems succeeded in accomplishing the tasks in the majority of times (**Table 2**).

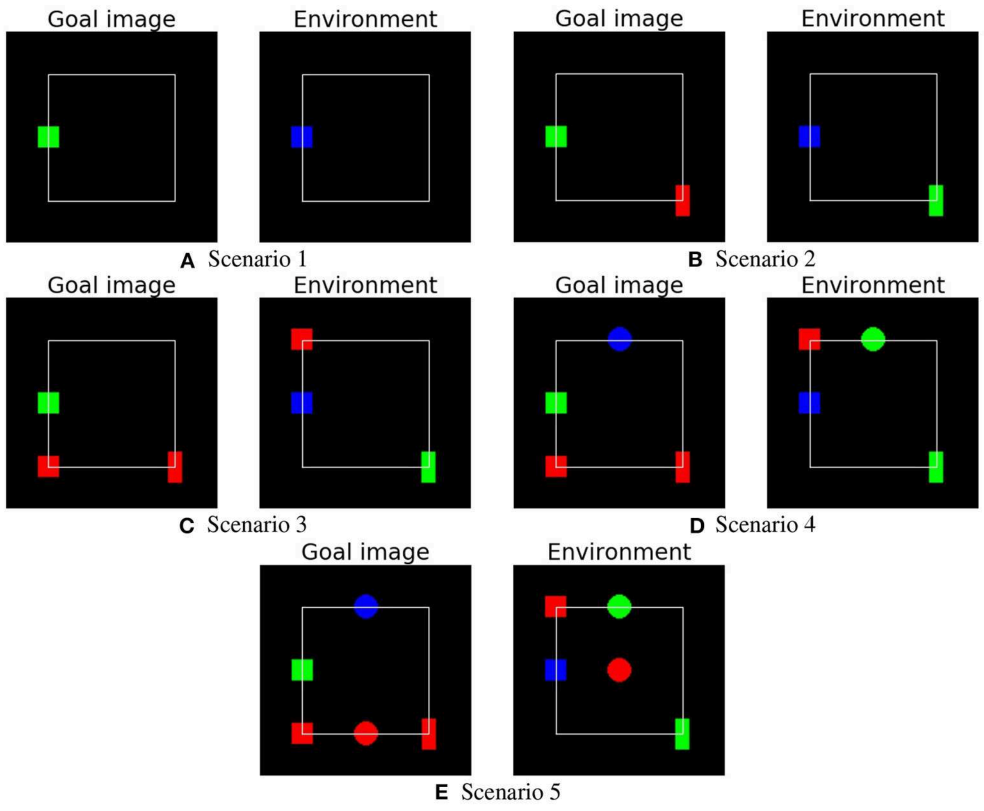


FIGURE 4 | The five different scenarios (goal image and initial environment setup) used to test the systems in the extrinsic phase. The scenarios involve an increasing number of objects. **(A)** Scenario 1. **(B)** Scenario 2. **(C)** Scenario 3. **(D)** Scenario 4. **(E)** Scenario 5.

TABLE 1 | Base test: affordance probabilities for all objects and actions.

Object		Move action prob.	Turn green prob.	Turn red prob.	Turn blue prob.
1	Red square	1.0	1.0	0.0	1.0
2	Green square	1.0	0.0	1.0	1.0
3	Blue square	1.0	1.0	1.0	0.0
4	Red circle	1.0	1.0	0.0	1.0
5	Green circle	1.0	0.0	1.0	1.0
6	Blue circle	1.0	1.0	1.0	0.0
7	Red rectangle	1.0	1.0	0.0	1.0
8	Green rectangle	1.0	0.0	1.0	1.0
9	Blue rectangle	1.0	1.0	1.0	0.0

Completion time in the IMP system showed an approximately quadratic dependency on the number of sub-goals (**Figure 6**) and also an increasing variance. The other two systems showed similar results. This test suggests that in a more complex environment with more objects, the system would be incapable of completing tasks within a reasonable amount of time. The reason for the poor scaling is mainly due to the simple bottom-up attention mechanism used here to guide attention, which uses a random exploration to find the objects needed to accomplish a certain sub-goal. A top-down mechanism capable of avoiding multiple explorations of the same objects would supposedly lead to a linear dependence of the completion time on the number of sub-goals.

3.1.2. Late-Object Tests

Three late-object tests were run. The features of the tests are summarized in **Table 3**. In the first test the six red-green-blue square and circle objects are present from the start of the simulation and have all affordances, while the red-green-blue rectangle objects are introduced late, have move affordance set to 0 (not movable) and the color affordances set to 1 ("greenable," "redable," and "blueable"). In the second test, the square objects are present from the start and afford all actions, the circle objects are present from the start and do not afford any action, and the three rectangle objects are introduced late and afford all actions. In the third test, the square and rectangle objects are present from the start and afford all actions, while the three circle objects are introduced late and do not afford any action. The three tests were run for 10,000 steps each and late objects were introduced after 2000 steps.

In the first late-object test, all three systems successfully learned the affordances of all objects, including the non-movable rectangles introduced late (**Figures 7–9**). After the intrinsic

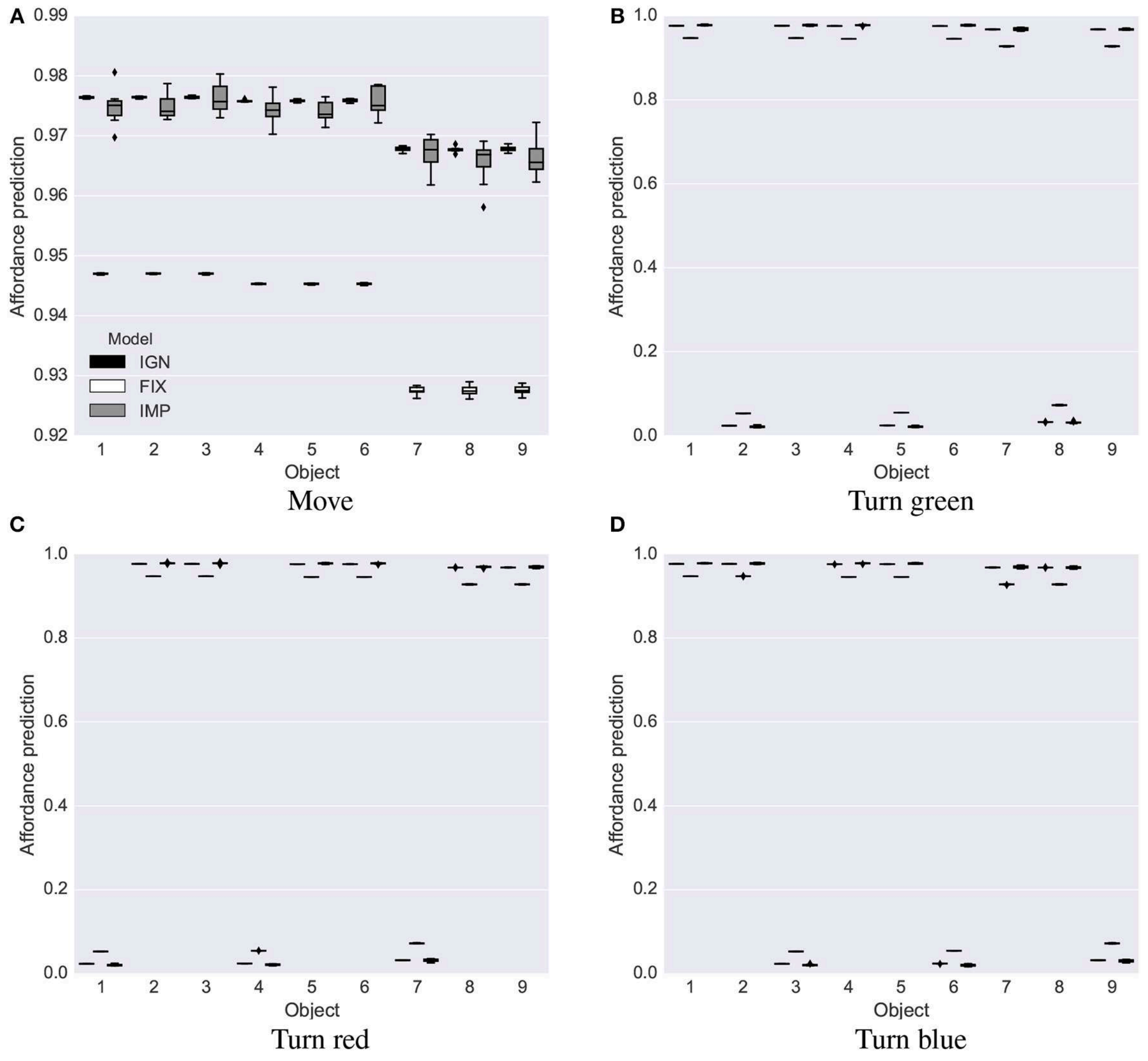


FIGURE 5 | Base test: affordance predictions (y-axis) after 6,000 learning steps for the four actions (four graphs) and nine objects (x-axis) averaged over 10 trials in the base test, for the IGN, FIX, and IMP systems. Mid-line of boxes shows median values, boxes show quartiles, and bars show the min-max range. The target values that the predictors had to estimate were 0 or 1. **(A)** Move. **(B)** Turn green. **(C)** Turn red. **(D)** Turn blue.

phase, the predictions of object affordances were correct for all three systems, except the "move" action affordances of the late objects 7 and 8 in the FIX system, which also showed the highest variance, followed by the IMP system (**Figure 10**).

Performance in the five extrinsic phase test scenarios (**Table 4**) was low for the FIX system compared to the IGN and IMP systems, and was highest for IMP.

In the second and third late-object test, the three systems differed in their behaviors while learning the affordances during the intrinsic phase, but all presented a similar performance when tested in the extrinsic phase, so we report the data related to them in **Supplementary Material**. In the second late-object test, the IGN and IMP systems first learned the affordances of objects introduced early in the simulations, and then focused and learned the affordances of the objects introduced late (**Supplementary Figures 1, 2**). Instead, the FIX system did not have such efficient focus (**Supplementary Figure 3**). After learning, the affordance predictions were correct for the IGN and IMP systems (with a higher variance for the IMP system) whereas the FIX system was less accurate and had a higher variance (**Supplementary Figure 4**).

Regarding the extrinsic-phase tests (**Table 5**), all three systems were successful in the first three scenarios, but failed in the fourth and fifth scenarios, showing that the different quality of affordances acquired in the intrinsic phase did not affect the performance in these particular tests. All systems failed the extrinsic phase scenario 4 and 5, involving an additional circle each, because in the late-object tests 2 and 3 the circle objects do not afford any action and so their state cannot be changed.

The first and second late-object tests confirm that the IGN and IMP systems outperform the FIX system in learning affordances as they can decide to explore a certain object on the basis of

TABLE 2 | Base test: success of the extrinsic-learning process for the three systems IGN, FIX, and IMP.

	Extrinsic-phase scenarios				
System	**1**	**2**	**3**	**4**	**5**
IGN	1.0	1.0	1.0	1.0	0.9
FIX	1.0	1.0	1.0	1.0	0.8
IMP	1.0	1.0	1.0	1.0	0.9

For each of the 10 intrinsic-phase runs and each of the 5 extrinsic-phase scenarios, the extrinsic-phase test was repeated 10 times: the figures reported here represent the portion of "successful" intrinsic-phase runs leading to > 90% successful extrinsic-phase tests.

a comparison between its expected information gain and the information gain expected on average from other objects.

In the third late-object test, none of the systems successfully learned to focus on, and predict accurately, the lack of affordances of the late circle objects (**Supplementary Figures 5–7**). As a consequence, after learning, the affordance predictions of such objects were inaccurate (far from 0) (object numbers 4, 5 and 6) and showed high variance for most objects (**Supplementary Figure 8**). This result can be explained by the fact that the predictions for the novel objects are bootstrapped from previously learned affordances of similar objects, in particular based on the color that causes synergies when it involves objects with same present/absent affordance, and interference in the opposite case. A mechanism of replay of past experience would possibly overcome this problem as it would intermix experience related to the different objects, allowing the neural-network predictors to disentangle the present/absent affordances of similar objects.

During the extrinsic phase (**Table 6**), all the three systems successfully accomplished the goal in the first three scenarios but not in the last two.

3.2. Stochastic Environment

3.2.1. Learning of Stochastic Affordances

The stochastic environment used stochastic affordances for some objects and actions whereas the other affordances were as in the deterministic environment (**Table 7**).

The intrinsic phase was run 10 times each for 10,000 steps. The plots show the average performance over 10 trials. The leaky average of the maximum utility estimation was updated with a leak rate $\nu = 0.1$ in both the IGN and IMP systems. The learning rate of the affordance predictors was set to $\alpha = 0.001$ and the learning rate of the learning-progress predictors was set to $\alpha = 0.0005$.

After learning, all three systems showed a good capacity to predict the affordances, but the IMP system was more accurate than the IGN and FIX systems as it could better employ the available learning time to accumulate more knowledge (**Figure 11**). In particular: (a) it learned better to estimate affordances, with a probability equal to 1.0 (**Figures 11A,C**) or 0.0 (**Figures 11B–D**); (b) it correctly learned the 0.8 probability for the change-color-to-blue action (**Figure 11D**).

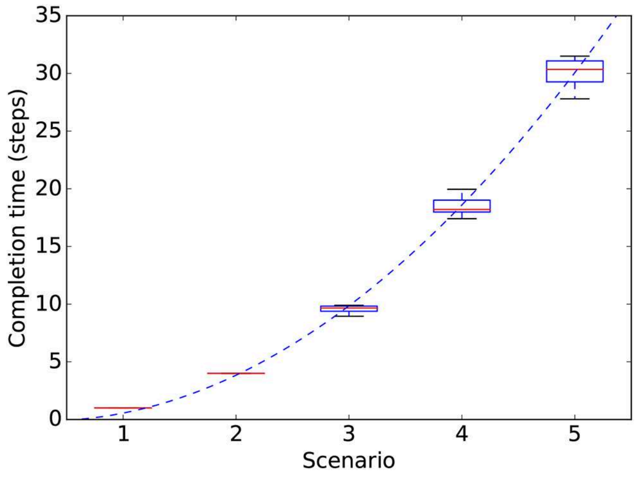


FIGURE 6 | Completion times (y-axis) for the IMP system in the different extrinsic-phase scenarios involving an increasing number of objects (x-axis). Data refers to 100 simulations (10 runs of the extrinsic-phase test for each of the 10 runs of the intrinsic-phase learning process). For each scenario, the mid-line of boxes shows median values, boxes show quartiles, and bars show the min-max range. The dashed line shows a quadratic fit: $y = ax^2 + bx = 1.364x^2 - 0.808x$.

TABLE 3 | The structure of the three late-object tests.

	Objects types:		
Late-test number	**Squares**	**Circles**	**Rectangles**
1	Start Move: 1 Color: 1	Start Move: 1 Color: 1	Late Move: 0 Color: 1
2	Start Move: 1 Color: 1	Start Move: 0 Color: 0	Late Move: 1 Color: 1
3	Start Move: 1 Color: 1	Late Move: 0 Color: 0	Start Move: 1 Color: 1

"Start/Late": objects that are present form the start of training/introduced later. "Move/Color": type of affordances. "1/0": presence/absence of the affordances.

Both the FIX and IGN systems fail to learn accurate affordance probabilities as they get high motivation signals for exploring stochastic objects even when there is no more knowledge to be gained on them. Only the IMP system is able to focus on different objects depending on the actual learning progress they can furnish.

As the IMP system learned affordance probabilities better than IGN and FIX, we ran the extrinsic-phase tests illustrated in the next section using only such a system.

3.2.2. Goal-Based Planning vs. Utility-Based Planning

The goal-based planner and the utility-based planner were compared by running an extrinsic-phase test in a stochastic environment with the four objects indicated in **Table 8**. The four

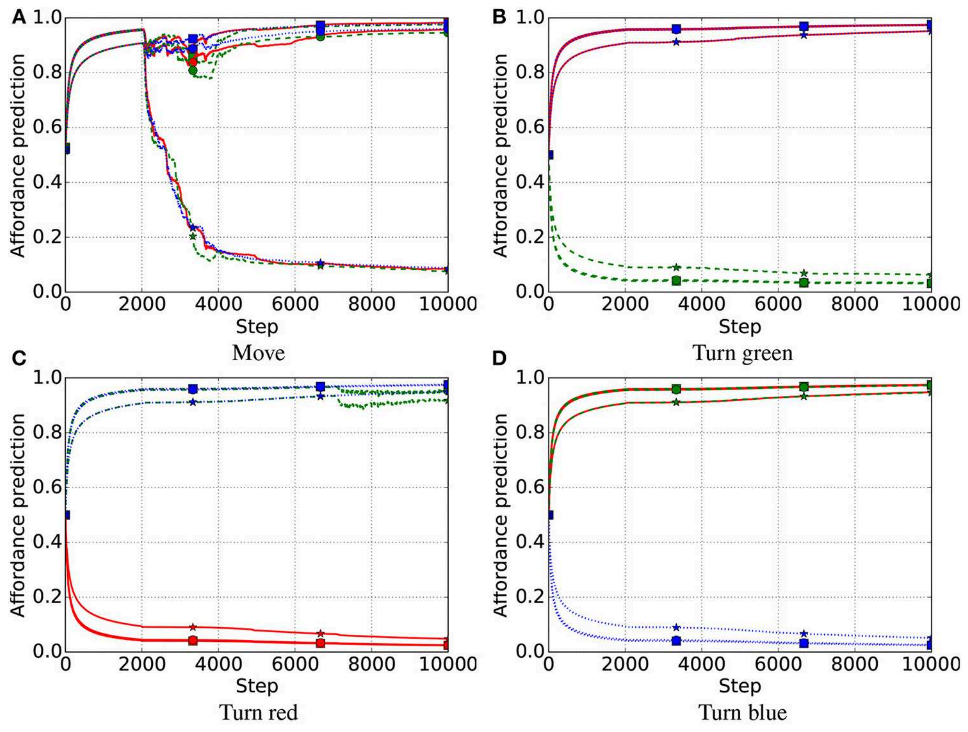


FIGURE 7 | First late-object test, IGN system. Affordance prediction for the four actions (4 graphs) and nine objects (lines in each graph) averaged over 10 trials. Red lines refer to red objects, green dashed lines to green objects, and blue dotted lines to blue objects. Markers on lines represent the shape of objects, where squares refer to square objects, circles to circular objects, and stars to rectangular objects. Note that an object of a color does not have the affordance to be turned to the same color (e.g., a red object cannot be turned red) as this involves no change. **(A)** Move. **(B)** Turn green. **(C)** Turn red. **(D)** Turn blue.

objects were assigned different values and the actions required for accomplishing each sub-goal had different probabilities of success (those corresponding to the affordances learned in the intrinsic phase). This resulted in a different expected utility of the objects.

The test was run 20 times for each of the 10 simulations of the intrinsic phase using different action budgets available to the system (1 to 5 actions). A small action-budget constraint introduces the need of deciding which actions to perform based on their expected utility.

The results show that the utility-based planner performed significantly better than the goal-based planner when it could rely on a small number of actions, and showed a statistical trend to do so for a higher number of actions (**Figure 12**). This shows an advantage of affordances for planning when the system knows the utility of different alternative sub-goals.

The smaller difference between the models in utility for a higher number of actions is expected due to the fact that if all goals can be accomplished, independently of their utility, the order of their accomplishment does not matter. However, reality, offering a very large number of alternative (sub-)goals with respect to the actions that can be performed, is similar to the case of the experiment where the system has only 1 or 2 actions available, so utility-based planning is very important in such conditions.

3.2.3. Learning of Forward Models in the Stochastic Environment

Having illustrated the utility-planning experiment, it is now possible to show that IMP outperformed IGN and FIX not only in terms of the quality of learned affordances but also in terms of the quality of the learned forward models. To this purpose, we compared the performance of the utility-planner using affordances and forward models trained with either one of the IGN/FIX/IMP mechanisms for 4,000 executed actions, a time not sufficient to fully learn the forward models. **Figure 13** shows the performance (overall gained utility) of the three utility planners using a maximum of 1, 2, or 3 actions and averaged over 100 repetitions of the experiment. The results show that IMP has a higher performance than IGN and FIX in all conditions; in particular, it is statistically better than FIX with 2 and 3 actions ($p < 0.05$), and better than IGN with 3 actions ($p < 0.05$).

The better performance of IMP could be due to either worse affordances or worse forward models of IGN and FIX. To ascertain this, we repeated the experiment using forward models trained for a time allowing convergence (10,000 executed actions) for all the three systems. In this case the three systems showed a similar performance (data not reported). This indicates that the better performance of IMP in the previous experiment was due to better forward models. A possible explanation of this is that there is a correlation between the difficulty of learning

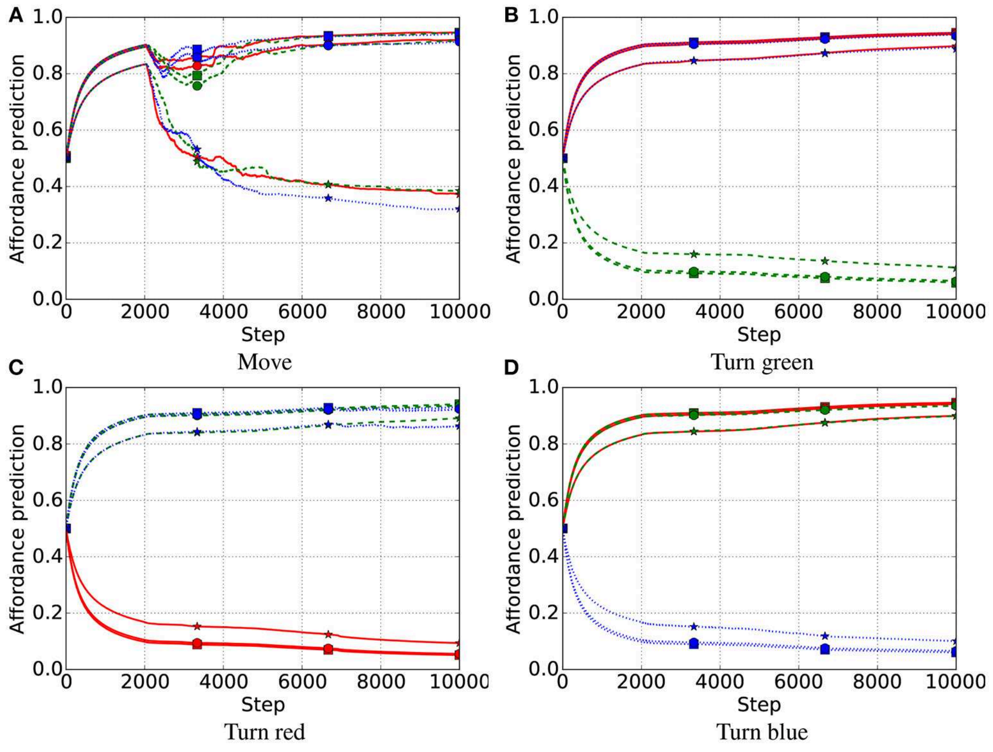


FIGURE 8 | First late-object test, FIX system. Red lines refer to red objects, green dashed lines to green objects, and blue dotted lines to blue objects; markers on lines represent the shape of objects, where squares refers to square objects, circles to circular objects, and stars to rectangular objects. **(A)** Move. **(B)** Turn green. **(C)** Turn red. **(D)** Turn blue.

the predictors estimating the affordance-probabilities and the predictors implementing the what-effect forward models as they share the same input (object image). So the effective decisions of IMP on which experiences to focus on to learn affordances also benefit the learning of the forward models. On the other hand, even if affordances of IGN and FIX have a lower quality than IMP (section 3.2.1), this does not negatively affect their performance as such lower quality does not impair the utility-based ranking of the object-related sub-goals.

3.2.4. Affordances Allow the Reduction of the Forward Planning Search

We have seen that a potential benefit of affordances for planning is the possibility of reducing the number of actions that should be checked during the generation of the forward trajectories. To validate this idea, we ran again the previous extrinsic-phase test (section 3.2.2) but without constraining the number of actions that the system could perform. In particular, we compared two systems, a first one checking all available actions and a second one restricting the forward-model-based search to only those actions having an affordance ≥ 0.5 (here this value excludes from the search all non-afforded actions). The results show that the use of affordances allows a significant reduction of the mean number of actions checked (**Figure 14**).

Consider that in realistic situations, the number of actions that can be performed in a certain condition is very high. Moreover, often several actions can be performed in sequences to accomplish a certain (sub-)goal, a situation not investigated here. In this case, the possible reduction based on affordances of the branching factor due to actions is even more important.

4. OTHER RELEVANT MODELS IN THE LITERATURE

The architecture presented here integrates functionalities that have been investigated in isolation in other computational systems. In this section we review the systems that are more closely related to the system presented here, and compare their main features.

Many works have focused on intrinsic motivations as a means to solving extrinsic challenging tasks where a long sequence of skills is required to solve a task or maximize a specific reward function ("sparse reward," e.g., Santucci et al., 2014). Instead, fewer works on open-ended learning have focused on intrinsic motivations for the autonomous acquisition of skills that are assembled in sequences in a later "extrinsic phase." For example, Schembri et al. (2007) and Kulkarni et al. (2016) present two reinforcement learning systems that undergo two separated learning phases. In the second extrinsic phase, both systems use reinforcement learning to solve complex tasks by assembling sequences of skills acquired in the first phase. In the

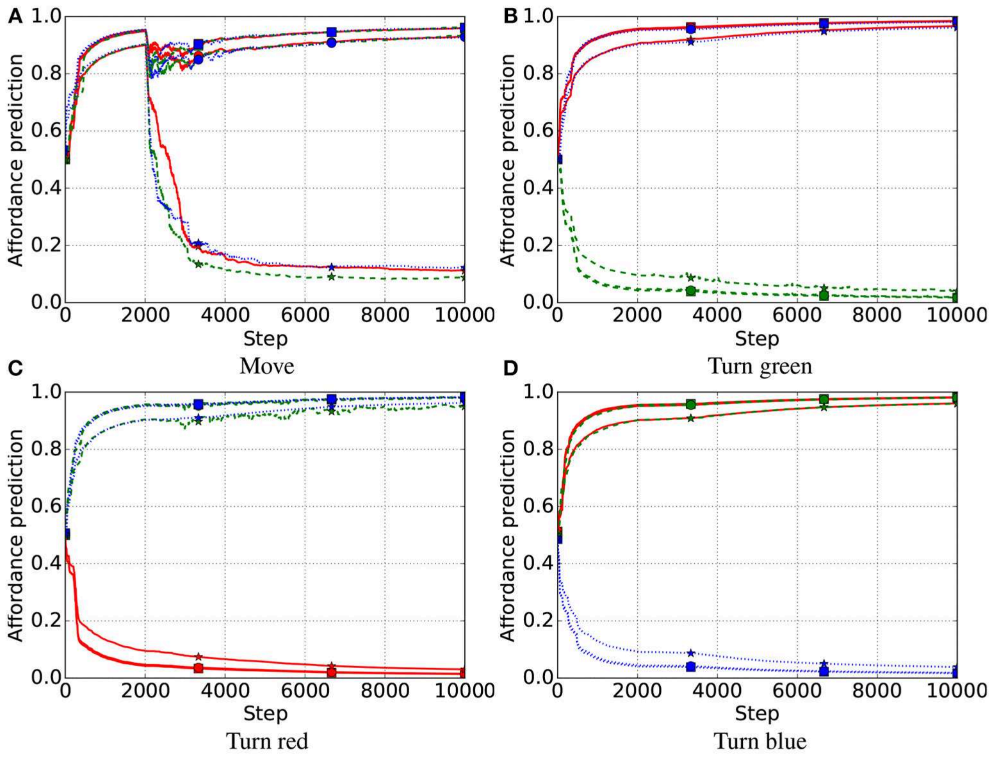


FIGURE 9 | First late-object test, IMP system. Affordance prediction for the four actions (4 graphs) and nine objects (lines in each graph) averaged over 10 simulations. Red lines refer to red objects, green dashed lines to green objects, and blue dotted lines to blue objects; markers on lines represent the shape of objects, where squares refers to square objects, circles to circular objects, and stars to rectangular objects. **(A)** Move. **(B)** Turn green. **(C)** Turn red. **(D)** Turn blue.

intrinsic phase, the first system learns the skills on the basis of reinforcement learning guided by intrinsic motivations and reward functions found by a genetic algorithm that uses the performance in the extrinsic phase as fitness. Instead, during the intrinsic phase the second system learns the skills on the basis of a mechanism generating skills when the agent's action causes "two objects to interact." Although these systems employ the idea of the two phases to develop and test open-ended learning systems, they do not investigate how they could learn affordances in the intrinsic phase and their possible use for planning.

Seepanomwan et al. (2017) investigates how a robot can exploit knowledge acquired with open-ended learning in an intrinsic phase to accomplish user-defined goals in a later extrinsic phase. In a first intrinsic phase the robot autonomously generates multiple "goals/outcomes" related to moving a ball in different positions on a table. In a following extrinsic phase the system reuses the acquired goals and skills to directly accomplish new goals (assigned to it by a user) or to more quickly learn the skills to do so. To this purpose, the system performs a one-step backward planning search by searching for the best skill to perform on the basis of the similarity of the user's goal with the goals of all skills. Contrary to here, the system can solve only simple but not compound tasks, can interact with only one object at a time since it does not have an attention system, and the initial condition is identical in each trial (the environment is "reset" after each skill performance).

Planning represents a central theme in artificial intelligence (Ghallab et al., 2004; Russell and Norvig, 2016). Here we only considered few aspects of planning relevant to face the issues related to open-ended learning and exploitation of affordances. Investigating the possible roles of affordances with planning, we have seen that when forward planning is used affordances allow the agent to short-list the available actions. Interestingly (cf. Russell and Norvig, 2016), until the late 90's the research on planning mainly focused on backward planning because this revealed more efficient than forward planning generating a wide search branching as many actions are applicable to each state. Later, forward planning became popular again, thanks to general-purpose heuristics that allowed the reduction of the search breadth based on domain-independent general heuristics, for example focusing only on the "positive effects" (usually denoting the action success) while ignoring the "delete effects" (usually involved in the violation of other sub-goals). This suggests that the common use of forward planning by organisms (Wikenheiser and Redish, 2015) might rely on affordances for pruning relevant actions: affordances hence are so important for organisms (Thill et al., 2013) because they not only support an efficient action but also planning.

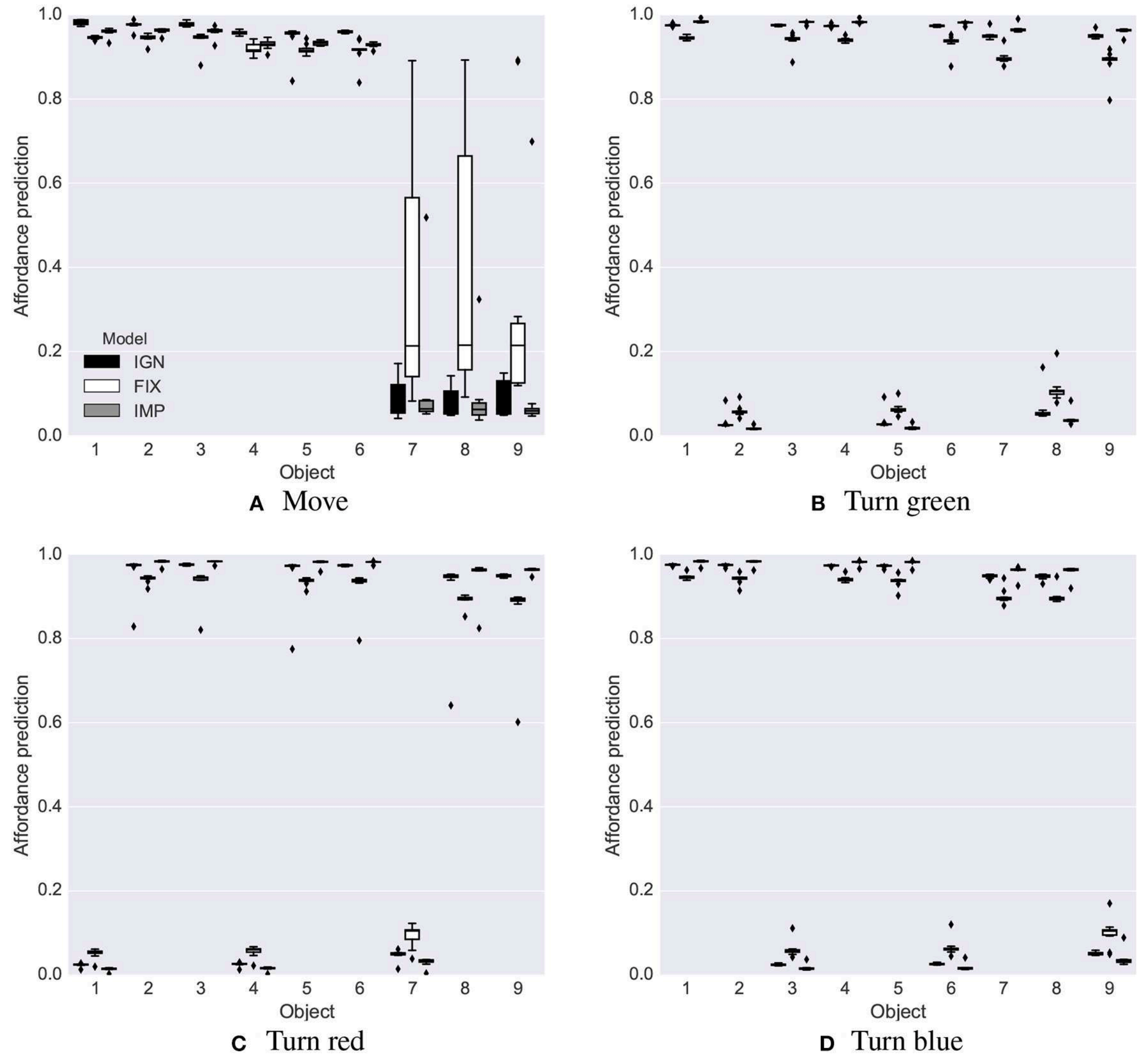


FIGURE 10 | First late-object test: affordance predictions after learning. Plotted as in **Figure 5**. **(A)** Move. **(B)** Turn green. **(C)** Turn red. **(D)** Turn blue.

The work presented here has multiple links with the autonomous/developmental robotics literature on affordances. In an early work, Stoytchev (2005) was among the first to propose the idea of a robot using affordances related to different objects (in this case tools) to evaluate their effects. Affordances, stored in a table, regarded the effects that actions could produce on different tools. Among other things, the work established the importance of focusing on objects rather than on the whole "state" of the environment for processing affordances.

Regarding the link between affordance acquisition and open-ended learning, Ugur et al. (2007) was among the first to use intrinsic motivations to support the acquisition of affordance knowledge. In particular, it investigated a mobile robot that had to learn to evaluate the "traversability" of a set of obstacle-objects in front of it. The robot scanned different possible directions of movement and decided to attempt to move along one of them if its ignorance with respect to the possible success in doing so was above a certain threshold. This aimed to invest the time and energy of the agent on learning the more uncertain affordances. Here we compared this mechanism with a more sophisticated mechanism where the ignorance for the current object and more uncertain affordance is compared with the estimated average ignorance for the other objects and affordances on which exploration time and energy might be alternatively invested.

The link between affordances and the possible relations between the elements of the object/action/effect triad was investigated by Montesano et al. (2008). The work equated affordances with the relations between such three elements and represented them with a Bayesian network within a probabilistic framework. Here we assumed a restricted definition of affordances, more closely linked to the initial definition, and this allowed us to investigate their relations with the elements involved in planning. Building on the probabilistic framework

TABLE 4 | First late-object test: success of the three systems IGN, FIX, and IMP in the five extrinsic scenarios.

	Extrinsic-phase scenario				
System	**1**	**2**	**3**	**4**	**5**
IGN	1.0	0.4	0.4	0.4	0.1
FIX	0.8	0.0	0.0	0.0	0.0
IMP	1.0	0.9	0.9	0.9	0.6

Data as in ***Table 2****.*

TABLE 5 | Second late-object test: success of the three systems IGN, FIX, and IMP in the five extrinsic scenarios.

	Extrinsic-phase scenario				
System	**1**	**2**	**3**	**4**	**5**
IGN	1.0	1.0	1.0	0.0	0.0
FIX	0.9	0.9	0.7	0.0	0.0
IMP	1.0	0.9	0.9	0.0	0.0

Data as in ***Table 2****.*

TABLE 6 | Third late-object test: success of the three systems IGN, FIX, and IMP in the five extrinsic scenarios.

	Extrinsic-phase scenario				
System	**1**	**2**	**3**	**4**	**5**
IGN	0.8	0.8	0.7	0.0	0.0
FIX	0.9	0.9	0.8	0.0	0.0
IMP	1.0	0.9	0.8	0.0	0.0

Data as in ***Table 2****.*

TABLE 7 | Stochastic environment: affordance probabilities for all objects and actions.

	Object	**Move action prob.**	**Turn green prob.**	**Turn red prob.**	**Turn blue prob.**
1	Red square	0.6	0.7	0.0	0.8
2	Green square	1.0	0.0	1.0	0.8
3	Blue square	1.0	0.7	1.0	0.0
4	Red circle	0.6	0.7	0.0	0.8
5	Green circle	1.0	0.0	1.0	0.8
6	Blue circle	1.0	0.7	1.0	0.0
7	Red rectangle	0.6	0.7	0.0	0.8
8	Green rectangle	1.0	0.0	1.0	0.8
9	Blue rectangle	1.0	0.7	1.0	0.0

of the work by Montesano et al. (2008), Gonçalves et al. (2014) propose to model affordances as the interaction between tools and objects based on their physical (geometrical) properties.

The link between affordances and planning was investigated in Ugur et al. (2009, 2011). The authors equated affordances to the forward models, in particular to the triad <object-features, action, effect>, where actions are pre-coded behaviors for moving or lifting objects and effects are clustered with a support vector machine. In a first phase the system learns the affordances and in a second phase the system is assigned a goal and plans the course of actions to pursue it based on a breadth-first forward search over actions and states until it finds a state similar to the goal. In Mar et al. (2015) a robot explored a pulling action performed with a rake-like tool to retrieve a target object. This allowed the robot to learn a mapping (through a support vector machine) between the pose of the tool in space and the affordances intended as the class (produced with a k-means clustering) of the possible "action parameters-retrieval effect" concatenated feature vector. For a given tool pose, this allowed the robot to select related affordance (action-effect class) and then to select the action parameters corresponding to the highest effect. This system shares some resemblance with the utility planner used here with the difference that in Mar et al. (2015) affordances support the selection of actions based on the amount of the expected desired (continuous) effect, whereas in our system they support the selection of actions based on their probability of producing the desired effect (which can be present/absent).

The affordance concept used here is analogous to the one of "preconditions" used in STRIPS-based planning *operators*. Preconditions establish if the operator is applicable or not to the current environment state (Fikes and Nilsson, 1972; Russell and Norvig, 2016). A similar concept is the "initiation set" of *options* in the reinforcement learning literature. The initiation set encompasses the states in which the action policy of the option can be performed (Sutton et al., 1999). Both concepts are deterministic. Instead, in utility-based reasoning actions produce desired effects only with a certain probability (Russell and Norvig, 2016). We think this action-success probability is the concept of artificial intelligence that is more similar to the concept of affordance used here.

Some of the relations between attention, affordances and intrinsic motivations were investigated in Nguyen et al. (2013) and Ivaldi et al. (2014). They proposed a robotic system endowed with a bottom-up attention system to detect the various objects available in the scene. Intrinsic motivations related to knowledge acquisition (3D object recognition) were used by the robot to decide which strategy to use (autonomous vs. social) to decide on which object to focus the learning resources. The work did not investigate, as here, the specific effects of a restricted focus of attention on the intrinsic-motivation mechanisms supporting affordance learning.

A last field of research related to this work involves active vision (Ballard, 1991). This literature is relevant as many action affordances tend to involve objects and a controllable visual sensor with a limited perceptual scope is a means to isolate information on specific objects (not considering here the important problem related to the fact that objects have different sizes and this requires an adjustable visual scope). Previous works (Ballard, 1991; Ognibene and Baldassare, 2015) have shown how such systems decrease the computational burden required by processing too wide visual images, and research on

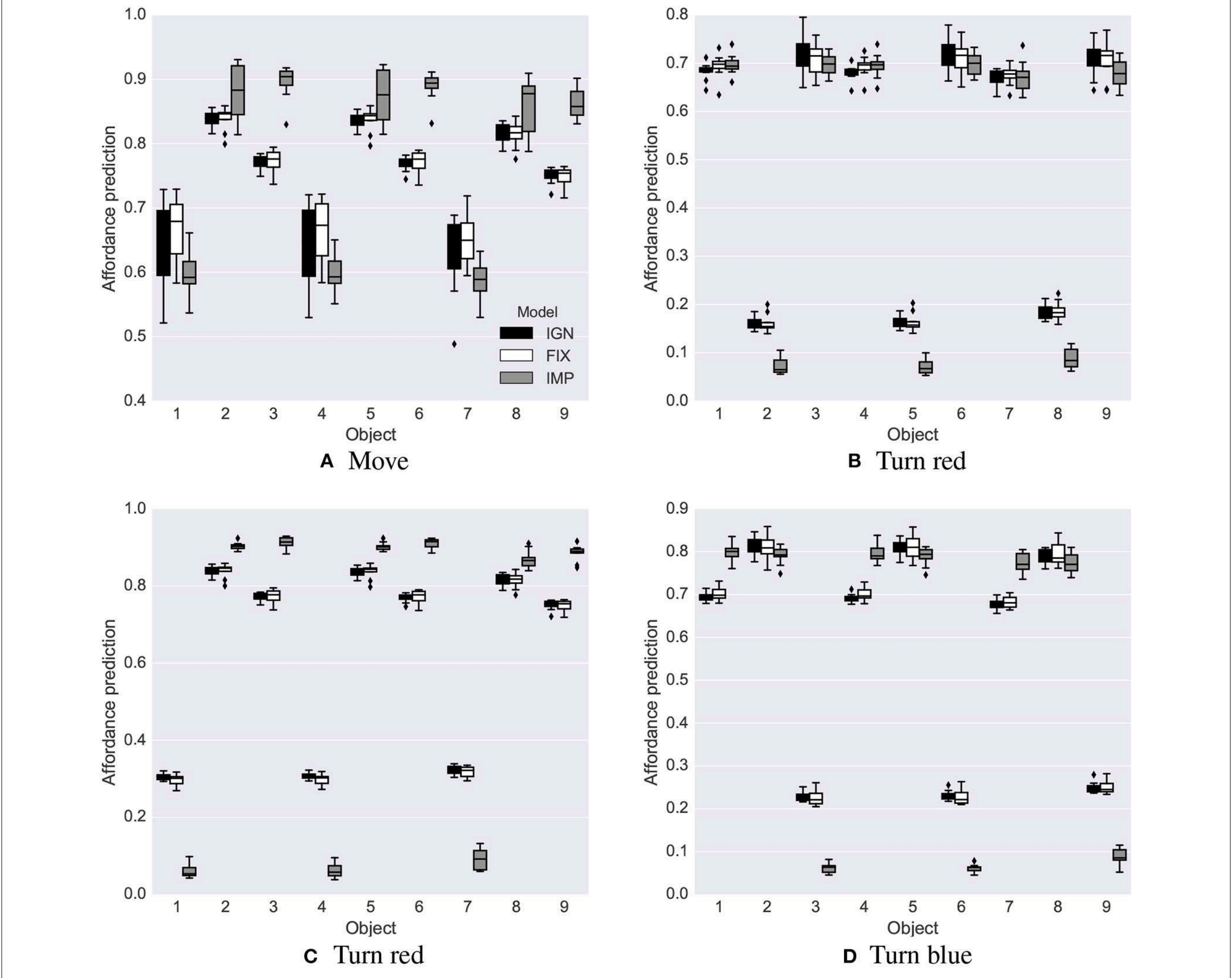


FIGURE 11 | Stochastic environment: affordance predictions after learning. Plotted as in **Figure 5**. **(A)** Move. **(B)** Turn red. **(C)** Turn red. **(D)** Turn blue.

TABLE 8 | Utility planning test: Objects in the utility planning test and their corresponding values, probability of goal-accomplishing action success and expected utility.

Object	Value	P(action success)	Expected utility
Blue circle	1	0.7	0.7
Green square	1	0.8	0.8
Red rectangle	2	0.7	1.4
Red square	4	0.6	2.4

state-of-the-art deep neural networks applied to vision problems is confirming the utility of an attentional focus (Xu et al., 2015). Here we use active vision in a different way to extract information on single objects, factored into information about the object state and its location in space. As mentioned in the introduction, this is a fundamental operation representing a first important step from a *factored* (featured-based) representation of the world state to a *structured* representation, allowing to reason on the state and relations between objects, typically used in classic planning (Russell and Norvig, 2016). Although this does not still allow the performance of the complex logic-based reasoning of classical planning, it allows the parsing of the whole goal into specific solvable object-centered sub-goals in the extrinsic phase.

We started to explore the factorization of a scene by an active vision system endowed with controllable restricted visual sensors in a camera-arm robot interacting with simple-shaped 2D objects as those used here (Ognibene et al., 2008, 2010). The sensor of this system was controlled not only with a bottom-up attention component, as here, but also by a top-down component able to learn to find a desired target-object by reinforcement learning: the latter component might be integrated into the current model in the future. The system was developed to accomplish only one extrinsic task, rather than multiple ones as here, and did not deal with open-ended learning. A system developed starting from the previous ones (Sperati and Baldassarre, 2014), again endowed

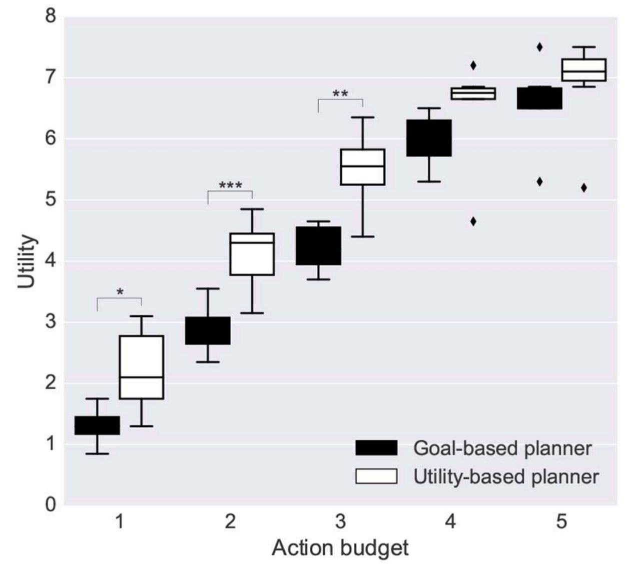


FIGURE 12 | Performance of the goal-based planner and the utility-based planner. Each bar is the average utility over 10 repetitions of the intrinsic-phase training and 20 runs of the extrinsic-phase test. Statistical significance was computed with a double-tailed t-test (Mid-line of boxes shows median values, boxes show quartiles, and bars show the min-max range, of the intrinsic-phase repetitions): $^*p < 0.05$; $^{**}p < 0.01$; and $^{***}p < 0.001$.

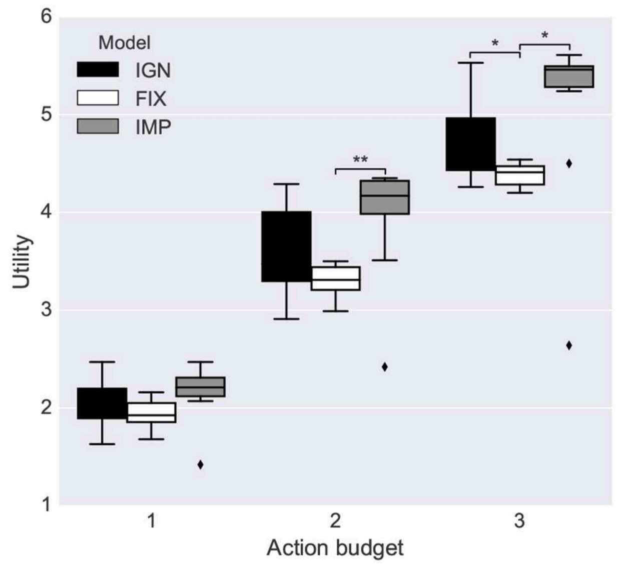


FIGURE 13 | Stochastic environment: quality of forward models acquired. Statistical significance is based on a double-tailed t-test (Mid-line of boxes shows median values, boxes show quartiles, and bars show the min-max range, of the ten intrinsic-phase repetitions). $^*p < 0.05$; $^{**}p < 0.01$.

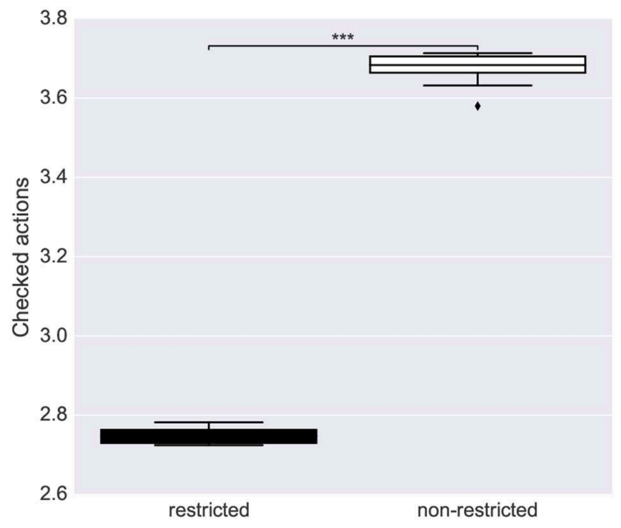


FIGURE 14 | Average number of actions to check for accomplishing each sub-goal in the case of affordance-based restricted and nonrestricted planning search. Statistical significance is based on a double-tailed t-test (Mid-line of boxes shows median values, boxes show quartiles, and bars show the min-max range, of the intrinsic-phase successful repetitions). $^{***}p < 0.001$.

with a bottom-up and a top-down attention component, is instead able to self-generate and learn tasks based on intrinsic motivations. This system has been further developed to self-generate and encode multiple visual-target *goals*, and to learn the camera-controller to find them (Sperati and Baldassarre, 2018). These systems have functionalities complementary with those of the system presented here, so they might be suitably integrated in the future to have a system able to: (a) self-generate goals and use them to drive the learning of the attention and motor skills to accomplish them through intrinsic motivations (previous systems) and (b) develop affordances and re-use the previously acquired skills to solve complex extrinsic tasks (system proposed here).

5. CONCLUSIONS AND FUTURE WORK

This work has focused on a possible specific instance of the concept of affordance intended as the probability of achieving a certain desired outcome associated to an action, by performing such action on a certain object. We investigated here three issues related to this concept: (a) within an open-ended autonomous learning context, how can intrinsic motivations guide affordance learning in a system that moves the attention of a visual sensor over different objects; (b) how can such an attention process support the decomposition of complex goals (tasks), involving multiple objects, into separated sub-goals related to single objects; (c) what could be the added value of affordances in planning systems already having sophisticated forward models of the world. For each issue we presented possible advancement with respect to the state of the art (section 4), and showed their advantages in specific experiments (section 3). Several aspects of the system could however be improved in future work.

Regarding the first issue, we proposed a mechanism to use intrinsic motivations (system IGN) to improve previously proposed ways (Ugur et al., 2007) with which a system endowed with a mechanism focusing on only one object/condition per time can decide whether or not to invest energy to explore it. The proposed mechanism is quite general and could in principle be used for any decision making process supported by a collection of information from the environment based on selective attention. The proposed solution is based on an adjustable variable storing the *opportunity cost* of the current choice, i.e., the value that the system looses by selecting the current option rather than alternative ones (Buchanan, 2008). The presented experiments support the effectiveness of this mechanism.

With respect to intrinsic motivations, in the case of deterministic scenarios where the system knows in advance that the affordance probability is either 0 or 1, the value of actions on the current object and the cost of alternatives was here estimated in terms of intrinsic motivations measuring the system ignorance (system IGN). This is not possible in stochastic scenarios where the affordance probability can be any value ranging in $(0, 1)$, so we proposed an intrinsic motivation tied to the improvement, rather than the level, of the probability estimation (system IMP). This solution, building on previous works on intrinsic motivations (e.g., Schmidhuber, 1991a; Santucci et al., 2013), led to a faster learning of affordance probabilities in our tests. However, an open problem of this solution, known in the literature (Santucci et al., 2013), is the fact that *error improvement* signals, as those used to compute intrinsic motivation in IMP, are small with respect to noise as they are equivalent to a derivative in time (vs. *error* signals used by IGN-like systems). This makes them more unstable: future work should face this problem.

Another important aspect related to autonomous learning driven by intrinsic motivations is that here, for the sake of focussing the research, the current system learns affordances on the basis of pre-wired actions and goals (expected outcomes of affordances). In a fully autonomous open-ended learning agent such actions and goals should instead be autonomously learned. Much literature has focused on the autonomous learning of actions and, more recently, of goals (e.g., Kulkarni et al., 2016; Santucci et al., 2016; Forestier et al., 2017; Cartoni and Baldassarre, 2018; Nair et al., 2018). Future work should thus aim to integrate the autonomous learning of affordances with the autonomous learning of actions and goals.

Regarding the second issue, related to the advantage for planning of having an attention system focusing on objects, we showed how the parsing of the scene into objects allows the solution of non-trivial planning problems on the basis of relatively simple one-step planning mechanisms. This agrees with previous proposals, such as the "object action compound" framework (Krüger et al., 2011; see also Montesano et al., 2008) stating the importance of representing information based on objects. Future work should investigate the advantage of object-centered attention and *multi-step* planning.

Although the introduction of focused visual sensors (attention) facilitates the parsing of the scene into objects, it also makes decision making more difficult. Indeed, the system has to look at different objects, and store information on them, to decide on which object to act or not. We have seen that the information to store can for example involve either the expected information gain, as requested by intrinsic motivations, or the utility of sub-goals, as requested by the solution of a utility-based problem. Here we have proposed a first solution to this problem that requires low computational resources (scanning objects in sequence, computing their expected utility, updating a variable that stores the maximum expected utility encountered this far, and deciding to act on the current object depending on how its utility compares with the maximum expected utility). This mechanism proved effective in tests. However, other more efficient (but also computationally more expensive) mechanisms could be used, in particular based on a memory of the specific utility of the different scanned objects. This information could be indexed by the different positions that have been visually inspected in the scene so that the information itself is readily usable to guide top-down attention processes and actions on specific target objects (Ognibene and Baldassare, 2015).

Regarding the third and last issue, related to the possible added value of affordances in planning systems, we showed that affordances as defined here can be useful in goal-based planning systems as they allow a search focused on actions that can be used in the current context. In section 4 we mentioned that this function is similar to that played by preconditions in STRIPS-based planning and by the "initiation set" in reinforcement-learning options. We have also seen that a second function that affordances can play, in particular for utility-based planning problems, is for weighting the importance of alternative goals based on the probability to accomplish them. The definition used to this purpose, $Pr(s'_{b,o} \in G|a, s_{b,o})$ can be related to one of *internal models* of the *transition function* $Pr(s'|a, s)$ used in model-based reinforcement-learning systems (Sutton and Barto, 2018). These models take into consideration stochastic environments rather than deterministic ones, as usually done in symbolic planning; but on the other hand they use *atomic representations*, s, of whole states, rather than information on single objects and their relations as done in symbolic planning using *structured representations* (Russell and Norvig, 2016). In this respect, the concept of affordances used here, pivoting on $s_{b,o}$, starts to integrate the two approaches as it focuses on single objects (the body and the target object), and at the same time it considers probabilities of their states (specifically encoded with *factored representations*, such as pixels images, as commonly done in robotic reinforcement learning models, Wiering and Van Otterlo, 2012). Future work could further develop this integration (e.g., see Konidaris et al., 2018).

Overall, we think that showing how attention can support a representation of information centered on objects rather than on whole states, and the implications of this for autonomous affordance learning and planning, is a very important issue to which this work contributed.

We conclude by discussing how the system might scale up to more complex scenarios. The overall architecture is expected to scale up well to more complex environments but the implementation of its components should be enhanced to such purpose. For the sake of simplicity here we developed the model components in a way that was sufficient to tackle a simple environment featuring a black background and non-overlapping objects. A realistic environment with a rich texture and several possibly-overlapping objects would produce cluttered images. To face this condition the system should be endowed with object segmentation capabilities (Zhang et al., 2008) or robust object recognition algorithms such as deep neural networks (LeCun et al., 2015). Interestingly, some of these latter algorithms have started to use attention mechanisms to improve object recognition (Maiettini et al., 2018): future work might investigate the links between these mechanisms and the attention processes presented here. More powerful deep learning models might also be used to implement the predictors used in the architecture. A last critical component is the simple bottom-up attention mechanism used to identify objects, and, as expected, this was limited (it scaled worse than linearly with the number of objects). The component could be enhanced with the addition of more sophisticated top-down attention mechanisms able to drive attention on the basis of the current knowledge on the identity and position of objects in the scene (Rasolzadeh et al., 2010; Sperati and Baldassarre, 2014, 2018; Ognibene and Baldassare, 2015).

A final general feature of the system that should be addressed in future work is the fact that the information flows between the several components of the architecture are managed by a hard-coded central algorithm using time flags and in some cases symbolic representations. This feature is shared by most architectures of this type. An alternative approach would be to follow the design of real brains where the information flows between components is continuous and has a distributed nature. An example of this is given in Baldassarre et al. (2013) proposing an architecture for goal-based open-ended learning where components are implemented on the basis of leaky-neuron neural networks. This strategy has the advantage of a higher biological realism (relevant when brain modeling is the research objective) and for having a higher tolerance to noise affecting the timing of events. On the other side it has the disadvantages of making it more difficult to tune the whole system and also to use some learning algorithms that require a precise timing of events.

AUTHOR CONTRIBUTIONS

GB: overall idea of the system, specification of the model and tests, analysis of results, and writing-up. WL: specification of the model and tests, implementation of the system, tests, data analysis, analysis of results, and writing-up. GG: specification of the model and initial tests, and analysis of results. VS: specification of the model and tests, analysis of results, and writing-up.

FUNDING

This project has received funding from the European Union Horizon 2020 Research and Innovation Programme, under Grant Agreement No. 713010, Project "GOAL-Robots – Goal-based Open-ended Autonomous Learning Robots."

ACKNOWLEDGMENTS

We thank Ilaria Maglione for support on the initial implementation of the system.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot.2019.00045/full#supplementary-material

REFERENCES

Asada, M., MacDorman, K. F., Ishiguro, H., and Kuniyoshi, Y. (2001). Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robot. Auton. Syst.* 37, 185–193. doi: 10.1016/S0921-8890(01)00157-9

Baldassarre, G. (2002). *Planning with neural networks and reinforcement learning.* (Ph.d. thesis). Computer Science Department, University of Essex, Colchester, United Kingdom.

Baldassarre, G. (2011). "What are intrinsic motivations? A biological perspective," in *Proceedings of the International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob-2011), 24-27 August* (Frankfurt am Main), E1–E8.

Baldassarre, G., Mannella, F., Fiore, V. G., Redgrave, P., Gurney, K., and Mirolli, M. (2013). Intrinsically motivated action-outcome learning and goal-based action recall: a system-level bio-constrained computational model. *Neural Netw.* 41, 168–187. doi: 10.1016/j.neunet.2012.09.015

Baldassarre, G., and Mirolli, M. (2013b). "Intrinsically motivated learning systems: an overview," in *Intrinsically Motivated Learning in Natural and Artificial Systems*, eds G. Baldassarre and M. Mirolli (Springer-Verlag, Berlin), 1–14.

Baldassarre, G., and Mirolli, M. (eds.). (2013a). *Intrinsically Motivated Learning in Natural and Artificial Systems*. Berlin; Heidelberg: Springer.

Ballard, D. H. (1991). Animate vision. *Artif. Intell.* 48, 57–86. doi: 10.1016/0004-3702(91)90080-4

Baranes, A., and Oudeyer, P. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robot. Auton. Syst.* 61, 49–73. doi: 10.1016/j.robot.2012.05.008

Barto, A., Mirolli, M., and Baldassarre, G. (2013). Novelty or surprise? *Front. Psychol.* 4:907. doi: 10.3389/fpsyg.2013.00907

Barto, A. G., Singh, S., and Chentanez, N. (2004). "Intrinsically motivated learning of hierarchical collections of skills," in *International Conference on Developmental Learning (ICDL2004)* (La Jolla, CA), 112–119.

Bratman, M. E. (1987). *Intentions, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press.

Buchanan, J. M. (2008). *Opportunity Cost, 2nd Edn*. London, UK: Macmillan Publishers.

Camoriano, R., Pasquale, G., Ciliberto, C., Natale, L., Rosasco, L., and Metta, G. (2017). Teaching robots to learn new objects in constant time. *arXiv:1605.05045v2*.

Cartoni, E., and Baldassarre, G. (2018). Autonomous discovery of the goal space to learn a parameterized skill. *arXiv 1805.07547v1*.

Castellini, C., Tommasi, T., Noceti, N., Odone, F., and Caputo, B. (2011). Using object affordances to improve object recognition. *IEEE Trans. Auton. Ment. Dev.* 3, 207–215. doi: 10.1109/TAMD.2011.2106782

Comoli, E., Coizet, V., Boyes, J., Bolam, J. P., Canteras, N. S., Quirk, R. H., et al. (2003). A direct projection from superior colliculus to substantia nigra for detecting salient visual events. *Nat. Neurosci.* 6, 974–980. doi: 10.1038/nn1113

Dauce, E. (2018). Fovea-based scene decoding through computationally-effective model-based prediction. *Front. Neurorobot.* 12:76. doi: 10.3389/fnbot.2018.00076

Fikes, R. E., and Nilsson, N. J. (1972). Strips: A new approach to the application of theorem proving to problem solving. *Artif. Intell.* 2, 189–208. doi: 10.1016/0004-3702(71)90010-5

Fitzpatrick, P., Metta, G., Natale, L., Rao, S., and Sandini, G. (2003). "Learning about objects through action-initial steps towards artificial cognition," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA03)*, Vol. 3 (Taipei: IEEE), 3140–3145.

Forestier, S., Mollard, Y., and Oudeyer, P.-Y. (2017). Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv 1708.02190v1*.

Gandhi, N. J., and Katnani, H. A. (2011). Motor functions of the superior colliculus. *Annu. Rev. Neurosci.* 34, 205–231. doi: 10.1146/annurev-neuro-061010-113728

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory and Practice*. Amsterdam: Elsevier.

Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Boston, MA: Houghton Mifflin.

Gonçalves, A., Saponaro, G., Jamone, L., and Bernardino, A. (2014). "Learning visual affordances of objects and tools through autonomous robot exploration," in *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on* (Espinho: IEEE), 128–133.

Ivaldi, S., Lyubova, N., Droniou, A., Padois, V., Filliat, D., Oudeyer, P.-Y., et al. (2014). Object learning through active exploration. *IEEE Trans. Auton. Ment. Dev.* 6, 56–72. doi: 10.1109/TAMD.2013.2280614

Konidaris, G., Kaelbling, L. P., and Lozano-Perez, T. (2018). From skills to symbols: learning symbolic representations for abstract high-level planning. *J. Artif. Intell. Res.* 61, 215–289. doi: 10.1613/jair.5575

Korf, R. E. (1985). Macro-operators: a weak method for learning. *Artif. Intell.* 26, 35–77. doi: 10.1016/0004-3702(85)90012-8

Krüger, N., Geib, C., Piater, J., Petrick, R., Steedman, M., Wörgötter, F., et al. (2011). Object–action complexes: grounded abstractions of sensory–motor processes. *Robot. Auton. Syst.* 59, 740–757. doi: 10.1016/j.robot.2011.05.009

Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., and Tenenbaum, J. B. (2016). Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. *arXiv 1604.06057*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521, 436–444. doi: 10.1038/nature14539

Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *Connect. Sci.* 15, 151–190. doi: 10.1080/09540090310001655110

Maiettini, E., Pasquale, G., Rosasco, L., and Natale, L. (2018). Speeding-up object detection training for robotics with falkon. *arXiv:1803.08740*. doi: 10.1109/IROS.2018.8593990

Mar, T., Tikhanoff, V., Metta, G., and Natale, L. (2015). "2d and 3d functional features for tool affordance learning and generalization on humanoid robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop Learning Object Affordances Fundamental Step Allow Prediction Planning Tool Use* (Hamburg).

Mirolli, M., and Baldassarre, G. (2013). "Functions and mechanisms of intrinsic motivations: the knowledge versus competence distinction," in *Intrinsically Motivated Learning in Natural and Artificial Systems*, eds G. Baldassarre and M. Mirolli (Berlin: Springer-Verlag), 49–72.

Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning object affordances: from sensory–motor coordination to imitation. *IEEE Trans. Robot.* 24, 15–26. doi: 10.1109/TRO.2007.914848

Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. (2018). "Visual reinforcement learning with imagined goals," in *The Second Lifelong Learning: A Reinforcement Learning Approach Workshop (LLRLA2018 at FAIM2018), number 1807.04742*. Stockholm.

Nguyen, S. M., Ivaldi, S., Lyubova, N., Droniou, A., Gerardeaux-Viret, D., Filliat, D., et al. (2013). "Learning to recognize objects through curiosity driven manipulation with the icub humanoid robot," in *IEEE International Conference on Development and Learning-Epirob* (Osaka).

Ognibene, D., and Baldassare, G. (2015). Ecological active vision: four bio-inspired principles to integrate bottom-up and adaptive top-down attention tested with a simple camera-arm robot. *IEEE Trans. Auton. Ment. Dev.* 7, 3–25. doi: 10.1109/TAMD.2014.2341351

Ognibene, D., Balkenius, C., and Baldassarre, G. (2008). "Integrating epistemic action (active vision) and pragmatic action (reaching): a neural architecture for camera-arm robots," in *From Animals to Animats 10: Proceedings of the Tenth International Conference on the Simulation of Adaptive Behavior (SAB2008)*, volume 5040 of *Lecture Notes in Artificial Intelligence, Osaka, Japan, 7-12 July 2008*, eds M. Asada, J. C. Hallam, J.-A. Meyer, and J. Tani (Berlin: Springer Verlag), 220–229.

Ognibene, D., Pezzulo, G., and Baldassarre, G. (2010). "How can bottom-up information shape learning of top-down attention-control skills?," in *Proceedings of 9th IEEE International Conference on Development and Learning (ICDL2010)* (Ann Arbor, MA), 231–237.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evol. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Oudeyer, P. Y., and Kaplan, F. (2007). What is intrinsic motivation? A typology of computational approaches. *Front. Neurorobot.* 1:6. doi: 10.3389/neuro.12.006.2007

Rasolzadeh, B., Björkman, M., Huebner, K., and Kragic, D. (2010). An active vision system for detecting, fixating and manipulating objects in the real world. *Int. J. Robot. Res.* 29, 133–154. doi: 10.1177/0278364909346069

Russell, S. J., and Norvig, P. (2016). *Artificial Intelligence: A Modern Approach, 3rd Edn*. Harlow: Pearson.

Ryan, R. M., and Deci, E. L. (2000). Intrinsic and extrinsic motivations: classic definitions and new directions. *Contemp. Educ. Psychol.* 25, 54–67. doi: 10.1006/ceps.1999.1020

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2012). "Intrinsic motivation mechanisms for competence acquisition," in *Proceeding of the IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob 2012), 7-9 November 2012* (San Diego, CA: IEEE), 1–6.

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2013). Which is the best intrinsic motivation signal for learning multiple skills? *Front. Neurorobot.* 7:22. doi: 10.3389/fnbot.2013.00022

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2014). "Cumulative learning through intrinsic reinforcements," in *Evolution, Complexity and Artificial Life*, eds S. Cagnoni, M. Mirolli, and M. Villani (Berlin: Springer), 107–122.

Santucci, V. G., Baldassarre, G., and Mirolli, M. (2016). Grail: a goal-discovering robotic architecture for intrinsically-motivated learning. *IEEE Trans. Cogn. Dev. Syst.* 8, 214–231. doi: 10.1109/TCDS.2016.2538961

Schembri, M., Mirolli, M., and Baldassarre, G. (2007). 'Evolving internal reinforcers for an intrinsically motivated reinforcement-learning robot," in *Proceedings of the 6th IEEE International Conference on Development and Learning (ICDL2007)* (Piscataway, NJ), 282–287.

Schmidhuber, J. (1991a). "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*. eds J. Meyer and S. Wilson (Boston, MA: MIT Press).

Schmidhuber, J. (1991b). "Curious model-building control systems," in *Proceedings of the International Joint Conference on Artificial Neural Networks*, Vol. 2 (Sydney, NSW), 1458–1463.

Seepanomwan, K., Santucci, V. G., and Baldassarre, G. (2017). "Intrinsically motivated discovered outcomes boost user's goals achievement in a humanoid robot," in *The Seventh Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob2017)*, eds J. Santos-Victor and G. Sandini (Lisbon: Istituto Superior Technico), 178–183.

Sperati, V., and Baldassarre, G. (2014). "Learning where to look with movement-based intrinsic motivations: a bio-inspired model," in *International Conferences on Development and Learning and Epigenetic Robotics (ICDL-Epirob)* (Genoa: IEEE), 461–468.

Sperati, V., and Baldassarre, G. (2018). A bio-inspired model learning visual goals and attention skills through contingencies and intrinsic motivations. *IEEE Trans. Cogn. Dev. Syst.* 10. doi: 10.1109/TCDS.2017.2772908

Stoytchev, A. (2005). "Behavior-grounded representation of tool affordances," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)* (Barcelona), 3060–3065.

Sutton, R. (1990). "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proceedings of the Seventh International Conference on Machine Learning*, Vol. 216 (Austin, TX), 216–224.

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction, 2nd Edn*. Cambridge, MA: The MIT Press.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.* 112, 181–211. doi: 10.1016/S0004-3702(99)00052-1

Sweeney, J. D., and Grupen, R. (2007). "A model of shared grasp affordances from demonstration," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on* (Pittsburgh, PA: IEEE), 27–35.

Thill, S., Caligiore, D., Borghi, A. M., Ziemke, T., and Baldassarre, G. (2013). Theories and computational models of affordance and mirror systems: an integrative review. *Neurosci. Biobehav. Rev.* 37, 491–521. doi: 10.1016/j.neubiorev.2013.01.012

Thrun, S., and Mitchell, T. (1995). Lifelong robot learning. *Robot. Auton. Syst.* 15, 25–46. doi: 10.1016/0921-8890(95)00004-Y

Ugur, E., Dogar, M. R., Cakmak, M., and Sahin, E. (2007). "Curiosity-driven learning of traversability affordance on a mobile robot," in *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on* (Piscataway, NJ: IEEE), 13–18.

Ugur, E., Oztop, E., and Sahin, E. (2011). Goal emulation and planning in perceptual space using learned affordances. *Robot. Auton. Syst.* 59, 580–595. doi: 10.1016/j.robot.2011.04.005

Ugur, E., and Piater, J. (2014). "Emergent structuring of interdependent affordance learning tasks," in *Proceedings of the Fourth Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob2014)*, eds G. Metta, M. Lee, and I. Fasel (New York, NY: Instituto Italiano di Tecnologia (IIT), IEEE), 481–486.

Ugur, E., Sahin, E., and Oztop, E. (2009). "Affordance learning from range data for multi-step planning," in *Proceedings of the Ninth International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, number 146 in Lund University Cognitive Studies*, eds L. Cañamero, P.-Y. Oudeyer, and C. Balkenius (Lund).

Ungerleider, L. G., and Haxby, J. V. (1994). 'what' and 'where' in the human brain. *Curr. Opin. Neurobiol.* 4, 157–165. doi: 10.1016/0959-4388(94)90066-3

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., et al. (2001). Autonomous mental development by robots and animals. *Science* 291, 599–600. doi: 10.1126/science.291.5504.599

White, R. W. (1959). Motivation reconsidered: the concept of competence. *Psychol. Rev.* 66:297. doi: 10.1037/h0040934

Wiering, M., and Van Otterlo, M. (2012). *Reinforcement Learning – State of the Art*, Vol. 12. Berlin: Springer.

Wikenheiser, A. M., and Redish, A. D. (2015). "Hippocampal sequences and the cognitive map," in *Analysis and Modeling of Coordinated Multi-neuronal Activity*, ed M. Tatsuno (Berlin: Springer), 105–129.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., et al. (2015). "Show, attend and tell: neural image caption generation with visual attention," in *International Conference on Machine Learning* (Lille), 2048–2057.

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: a survey. *ACM Comput. Surv.* 38:13. doi: 10.1145/1177352.1177355

Zhang, H., Fritts, J. E., and Goldman, S. A. (2008). Image segmentation evaluation: a survey of unsupervised methods. *Comput. Vis. Image Understand.* 110, 260–280. doi: 10.1016/j.cviu.2007.08.003

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.