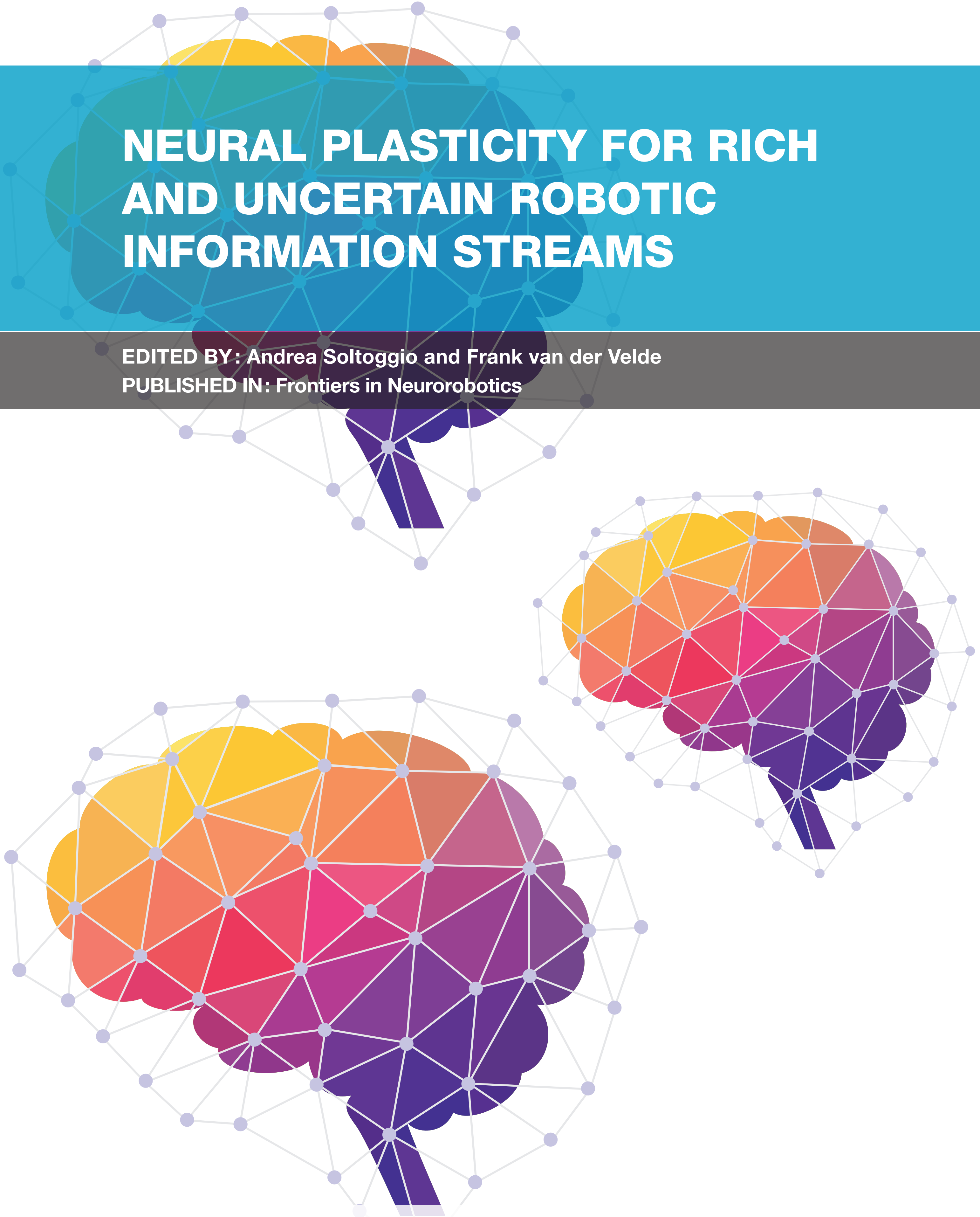
NEURAL PLASTICITY FOR RICH AND UNCERTAIN ROBOTIC INFORMATION STREAMS

EDITED BY: Andrea Soltoggio and Frank van der Velde

PUBLISHED IN: Frontiers in Neurorobotics



frontiers Research Topics

ISSN 1664-8714
ISBN 978-2-88919-995-2
DOI 10.3389/978-2-88919-995-2

About Frontiers

Frontiers is more than just an open-access publisher of scholarly articles: it is a pioneering approach to the world of academia, radically improving the way scholarly research is managed. The grand vision of Frontiers is a world where all people have an equal opportunity to seek, share and generate knowledge. Frontiers provides immediate and permanent online open access to all its publications, but this alone is not enough to realize our grand goals.

Frontiers Journal Series

The Frontiers Journal Series is a multi-tier and interdisciplinary set of open-access, online journals, promising a paradigm shift from the current review, selection and dissemination processes in academic publishing. All Frontiers journals are driven by researchers for researchers; therefore, they constitute a service to the scholarly community. At the same time, the Frontiers Journal Series operates on a revolutionary invention, the tiered publishing system, initially addressing specific communities of scholars, and gradually climbing up to broader public understanding, thus serving the interests of the lay society, too.

Dedication to Quality

Each Frontiers article is a landmark of the highest quality, thanks to genuinely collaborative interactions between authors and review editors, who include some of the world's best academicians. Research must be certified by peers before entering a stream of knowledge that may eventually reach the public - and shape society; therefore, Frontiers only applies the most rigorous and unbiased reviews.
Frontiers revolutionizes research publishing by freely delivering the most outstanding research, evaluated with no bias from both the academic and social point of view.
By applying the most advanced information technologies, Frontiers is catapulting scholarly publishing into a new generation.

What are Frontiers Research Topics?

Frontiers Research Topics are very popular trademarks of the Frontiers Journals Series: they are collections of at least ten articles, all centered on a particular subject. With their unique mix of varied contributions from Original Research to Review Articles, Frontiers Research Topics unify the most influential researchers, the latest key findings and historical advances in a hot research area! Find out more on how to host your own Frontiers Research Topic or contribute to one as an author by contacting the Frontiers Editorial Office: **researchtopics@frontiersin.org**

NEURAL PLASTICITY FOR RICH AND UNCERTAIN ROBOTIC INFORMATION STREAMS

Topic Editors:
Andrea Soltoggio, Loughborough University, UK
Frank van der Velde, University of Twente, Netherlands

Citation: Soltoggio, A., van der Velde, F., eds. (2016). Neural Plasticity for Rich and Uncertain Robotic Information Streams. Lausanne: Frontiers Media. doi: 10.3389/978-2-88919-995-2

Table of Contents

Editorial: Neural plasticity for rich and uncertain robotic information streams

Andrea Soltoggio[1]* *and Frank van der Velde*[2]

[1] Computer Science, Loughborough University, Loughborough, UK, [2] CTIT, CPE-BMS, University of Twente, Enschede, Netherlands

Keywords: neural plasticity, neural adaptation, neuro-robotics, cognitive modeling, emobodied cognition

OPEN ACCESS

Edited by:
Florian Röhrbein,
Technische Universität München,
Germany

Reviewed by:
Georg Martius,
Institute for Science and Technology
Austria, Austria

****Correspondence:***
Andrea Soltoggio
a.soltoggio@lboro.ac.uk

Received: *03 September 2015*
Accepted: *08 October 2015*
Published: *27 October 2015*

Citation:
Soltoggio A and van der Velde F (2015) Editorial: Neural plasticity for rich and uncertain robotic information streams. Front. Neurorobot. 9:12. doi: 10.3389/fnbot.2015.00012

Models of adaptation and neural plasticity are often demonstrated in robotic scenarios with heavily pre-processed and regulated information streams to provide learning algorithms with appropriate, well timed, and meaningful data to match the assumptions of learning rules. On the contrary, natural scenarios are often rich of raw, asynchronous, overlapping and uncertain inputs and outputs whose relationships and meaning are progressively acquired, disambiguated, and used for further learning. Therefore, recent research efforts focus on neural embodied systems that rely less on well timed and pre-processed inputs, but rather extract autonomously relationships and features in time and space. The bio-inspired focus does not seek the most effective machine learning method to solve those problems, it rather points toward a better understanding of problem solving mechanisms in neural systems, which can in turn also provide viable solutions to difficult problems.

Realistic models of plasticity must account for delayed rewards (Soltoggio et al., 2013a), noisy and ambiguous data (Soltoggio et al., 2013b), and emerging and novel input features during online and value learning (Krichmar and Röhrbein, 2013). Those factors have indeed been an emerging focus of search (e.g., Sporns and Alexander, 2003; Lungarella and Sporns, 2006; Martius et al., 2013), with a growing number of studies that cannot be reviewed in this short editorial. Such approaches model the progressive acquisition of knowledge by neural systems through experience in environments that may be affected by ambiguities, uncertain signals, delays, or novel features (Pugh et al., 2014; Soltoggio, 2015). This Research Topic in Frontiers in Neurorobotics explored fundamental properties and dynamics of neural learning systems that are naturally immersed in a rich information flow. We are pleased with the contributions collected in this Research Topic, each of which addresses key topics in this emerging and important field of research.

One overarching problem in this field is that of making sense of large amounts of data from sensory systems in order to recognize particular situations and perform basic tasks. Parisi and colleagues took a self-organizing neural approach to action recognition using human pose-motion features. The Growing When Required (GWR) networks manifest a high-level structural plasticity that regulates network complexity in relation to the task (Parisi et al., 2015). Such a bio-inspired approach recorded state-of-the-art performance on a dataset of full-body actions captured with a depth sensor, with competitive results in a public benchmark of domestic daily actions.

Another source of large, noisy and uncertain data is found in robotic tactile sensors. Chou et al. (2015) deployed a specific robot called CARL-SJR with a full-body tactile sensory area. CARL-SJR encourages people to communicate with it through gentle touch, and provides feedback to users by displaying bright colors on its surface. The time-delayed and uncertain nature of the interactions poses challenges to the formation of correct associations between stimuli, rewards and actions. The approach devised by Chou et al. (2015) experiments with a strongly bio-inspired architecture of spiking neurons with neuromodulated plasticity. The model abstracts brain areas such as the primary somatosensory cortex, prefrontal cortex, striatum, and the insular cortex to process noisy data generated directly from CARL-SJR's tactile sensory area. The result is a robust learning mechanism that reliably forms correct associations and preferences for directions without heavily pre-processed inputs.

Uncertainty and large amount of data are also found in collaborative multi-robot scenarios in which multiple robots work alongside humans. Galbraith and colleagues propose a motor babbling approach to learn a complex set of relations and interactions with the 11-degrees-of-freedom RoPro Calliope mobile robot (Galbraith et al., 2015). Motor babbling of its wheels and arm enabled the Calliope to learn how to relate visual and proprioceptive information to achieve hand-eye-body coordination.

Motor control is a problem in which neural plasticity results in high level of adaptation, adjusting neural systems to operate in combination with specific bio-mechanical structures and morphologies. (Burms et al., 2015) demonstrated the utility of modulated Hebbian plasticity in embodied computation for compliant robotics. In such scenarios, control policies are generally unknown due to the partial offload of control policies to morphological computation. Modulated Hebbian plasticity was shown to lead to hybrid controllers that naturally integrate the computations that are performed by the robot's body into a neural network architecture. Those results demonstrate the potential of universal applicability of plasticity rules to complex control problems.

A similar problem was tackled in Dasgupta et al. (2015) in which they used distributed recurrent neural networks with synaptic adaptation to find a range of complex behaviors for walking robots. In particular, their approach demonstrated a remarkable flexibility in designing control systems that can work with multi-legged robots. A Central Pattern Generator is used to feed a self-adaptive reservoir network, which in turn provides motor control through a read-out integration unit. These results contribute to demonstrate the efficacy and continuous advancement of plastic neural models in complex input-output control scenarios.

The overall vision provided by these research papers outlines an increasingly more effective deployment of plastic neural models to tackle complex perception and control problems in which noise, uncertainty and delays pose a challenge to many algorithms. This vision matches the intuition of bioinspired neurorobotics approaches that propose advanced, plastic neural systems as viable models when sensory-motor information flows approach the richness and complexity found in the behavior of biological systems. We foresee a continuous growing attention to this emerging research area, in particular related to the development of more effective, scalable and general neural learning algorithms to effectively tackle rich and uncertain robotic information streams.

AUTHOR CONTRIBUTIONS

AS devised the structure. Both authors AS and FV formulated the content and wrote the paper.

REFERENCES

Burms, J., Caluwaerts, K., and Dambre, J. (2015). Reward modulated hebbian plasticity as leverage for partially embodied control in compliant robotics. *Front. Neurorobot.* 9:9. doi: 10.3389/fnbot.2015.00009

Chou, T.-S., Bucci, L. D., and Krichmar, J. L. (2015). Learning touch preferences with a tactile robot using dopamine modulated stdp in a model of insular cortex. *Front. Neurorobot.* 9:6. doi: 10.3389/fnbot.2015.00006

Dasgupta, S., Goldschmidt, D., Wörgötter, F., and Manoonpong, P. (2015). Distributed recurrent neural forward models with synaptic adaptation and CPG-based control for complex behaviors of walking robots. *Front. Neurorobot.* 9:10. doi: 10.3389/fnbot.2015.00010

Galbraith, B. V., Guenther, F. H., and Versace, M. (2015). A neural network-based exploratory learning and motor planning system for co-robots. *Front. Neurorobot.* 9:7. doi: 10.3389/fnbot.2015.00007

Krichmar, J. L., and Röhrbein, F. (2013). Value and reward based learning in neurorobots. *Front. Neurorobot.* 7:13. doi: 10.3389/fnbot.2013.00013

Lungarella, M., and Sporns, O. (2006). Mapping information flow in sensorimotor networks. *PLoS Comput. Biol.* 2:e144. doi: 10.1371/journal.pcbi.0020144

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

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

Pugh, J. K., Soltoggio, A. and Stanley, K. O. (2014). "Real-time hebbian learning from autoencoder features for control tasks," in *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE XIV)* (Cambridge, MA: MIT Press), 202–209.

Soltoggio, A. (2015). Short-term plasticity as cause-effect hypothesis testing in distal reward learing. *Biol. Cybern.* 109, 75–94.

Soltoggio, A., Lemme, A., Reinhart, F. R., and Steil, J. J. (2013a). Rare neural correlations implement robotic conditioning with reward delays and disturbances. *Front. Neurorobot.* 7:6. doi: 10.3389/fnbot.2013.00006

Soltoggio, A., Reinhart, F. R., Lemme, A., and Steil, J. J. (2013b). "Learning the rules of a game: neural conditioning in human-robot interaction with delayed rewards," in *Proceedings of the Third Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics* (Osaka).

Sporns, O., and Alexander, W. H. (2003). "Neuromodulation in a learning robot: interactions between neural plasticity and behavior," in *Neural Networks, 2003. Proceedings of the International Joint Conference on,* Vol. 4 (IEEE), 2789–2794. doi: 10.1109/IJCNN.2003.1224010

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 June 2015
doi: 10.3389/fnbot.2015.00003

Self-organizing neural integration of pose-motion features for human action recognition

German I. Parisi *, Cornelius Weber and Stefan Wermter*

Department of Informatics, Knowledge Technology Institute, University of Hamburg, Hamburg, Germany

The visual recognition of complex, articulated human movements is fundamental for a wide range of artificial systems oriented toward human-robot communication, action classification, and action-driven perception. These challenging tasks may generally involve the processing of a huge amount of visual information and learning-based mechanisms for generalizing a set of training actions and classifying new samples. To operate in natural environments, a crucial property is the efficient and robust recognition of actions, also under noisy conditions caused by, for instance, systematic sensor errors and temporarily occluded persons. Studies of the mammalian visual system and its outperforming ability to process biological motion information suggest separate neural pathways for the distinct processing of pose and motion features at multiple levels and the subsequent integration of these visual cues for action perception. We present a neurobiologically-motivated approach to achieve noise-tolerant action recognition in real time. Our model consists of self-organizing Growing When Required (GWR) networks that obtain progressively generalized representations of sensory inputs and learn inherent spatio-temporal dependencies. During the training, the GWR networks dynamically change their topological structure to better match the input space. We first extract pose and motion features from video sequences and then cluster actions in terms of prototypical pose-motion trajectories. Multi-cue trajectories from matching action frames are subsequently combined to provide action dynamics in the joint feature space. Reported experiments show that our approach outperforms previous results on a dataset of full-body actions captured with a depth sensor, and ranks among the best results for a public benchmark of domestic daily actions.

Keywords: action recognition, visual processing, depth information, neural networks, self-organizing learning, robot perception

OPEN ACCESS

Edited by:
Andrea Soltoggio,
Loughborough University, UK

Reviewed by:
Diego Faria,
University of Coimbra, Portugal
Stephen Marsland,
Massey University, New Zealand

****Correspondence:***
German I. Parisi,
Department of Informatics, Knowledge Technology Institute, University of Hamburg, Vogt-Koelln-Strasse 30, Hamburg 22527, Germany
parisi@informatik.uni-hamburg.de

Received: *02 March 2015*
Accepted: *20 May 2015*
Published: *09 June 2015*

Citation:
Parisi GI, 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

1. Introduction

For humans and other mammals, the recognition of others' actions represents a crucial ability underlying social interaction and perceptual decision-making. Similarly, the visual recognition of complex movements may be fundamental for artificial systems to enable natural human-robot interaction (HRI) and action-driven social perception (Layher et al., 2012). The robust classification of full-body, articulated actions represents a key component of assistive robots aiming to provide reliable recognition of user behavior and remains an enticing milestone for artificial systems embedded in socially-aware agents (Kachouie et al., 2014). When operating in complex environments, algorithms for the visual recognition of action classes face a trade-off between satisfactory accuracy and minimal recognition latency (Ellis et al., 2013). There exists a vast set of challenges to be addressed regarding the efficient processing of raw visual information and the generalization of actions for effective inter-class discrimination, while neglecting subtle intra-class differences. Moreover, an enduring bottleneck for vision-based approaches regards the segmentation of human shape and motion from 2D image sequences, often constrained in terms of computational efficiency and robustness to illumination changes (Weinland et al., 2011).

In the last half decade, the use of low-cost depth sensing devices such as the Microsoft Kinect and ASUS Xtion has led to a great number of vision-based applications using depth information instead of, or in combination with, brightness and color information (for a review see Han et al., 2013). This sensor technology provides depth measurements used to obtain reliable estimations of 3D human motion in cluttered environments, including a set of body joints in real-world coordinates and limb orientations. Despite recent research efforts combining 3D skeleton models with machine learning and neural network approaches, the question remains open on how to better process extracted body features for effectively learning the complex dynamics of actions in real-world scenarios. For instance, in such scenarios the correct classification of actions may be hindered by noisy and missing body joints caused by systematic sensor errors or temporary occluded body parts (Parisi and Wermter, 2013). Nevertheless, a robust, noise-tolerant system should also operate under such adverse conditions. A promising scheme to tackle this demanding task is the implementation of computational models built upon evidence from the biological visual system. This scheme is supported by the fact that human observers are capable of carrying out action discrimination effortlessly (Blake and Shiffrar, 2007), outperforming artificial systems. In particular, neural mechanisms underlying action recognition have been broadly studied in the literature (Perret et al., 1982; Giese and Poggio, 2003), thereby encompassing multidisciplinary research to shed light on perceptual representations and neural pathways responsible for triggering robust action perception in humans and non-human primates. Simplified models of brain areas processing visual cues have been adopted as a stepping stone to numerous artificial systems dealing with the detection and classification of articulated, complex motion such as human actions (Giese and Poggio, 2003; Layher et al., 2012).

In this work, we present a learning architecture for the recognition of actions based on the following three assumptions consistent with neurobiological evidence from the mammalian visual system: (1) Complex motion is analyzed in parallel by two separated pathways and subsequently integrated to provide a joint percept (Perret et al., 1982; Vangeneugden et al., 2009); (2) Both channels contain hierarchies to extrapolate shape and optic-flow features with increasing complexity (Giese and Poggio, 2003), from low- to high-level representations of the visual stimuli; (3) Input-driven self-organization is crucial for the cortex to tune the neurons according to the distribution of the inputs (von der Malsburg, 1973; Kohonen, 1993; Miikkulainen et al., 2005). Under these assumptions, we carry out action learning and classification through a two-pathway hierarchy of growing self-organizing networks that cluster separately pose and motion samples. During the training, Growing When Required networks (Marsland et al., 2002) dynamically change their topological structure through competitive Hebbian learning (Martinetz, 1993) to incrementally match the input space. The learning process is built upon input-driven synaptic plasticity (Pascual-Leone et al., 2011) and habituation (Thompson and Spencer, 1966). Clustered neuronal activation trajectories from the parallel pathways are subsequently integrated to generate prototype neurons representing action dynamics in the joint pose-motion domain, resembling the neural integration of multi-cue action features in the visual cortex (Beauchamp et al., 2003).

In previous research we explored the use of hierarchical self-organization for integrating pose-motion cues using Growing Neural Gas (GNG) learning (Parisi et al., 2014a,c). The unsupervised learning algorithm was extended with two labeling functions for classification purposes. In this work, we use GWR networks that can create new neurons whenever the activity of the best neuron matching the input is not sufficiently high, leading to a more efficient convergence with respect to GNG networks that use a fixed insertion interval. In the previous model, an extra network was used to automatically detect outliers in the training and test set. However, the removal of noisy cues via an additional specialized network lacks neurobiological support and adds complexity to the model. With the use of an extended GWR learning mechanism, we will show that this process can be embedded naturally into the self-organizing hierarchy for the clustering of action cues and allows to remove noisy samples also during live classification.

The rest of this paper is structured as follows. In Section 2, we introduce biological evidence and models for neural integration of multiple visual cues and present an overview of state-of-the-art learning approaches for human action recognition using depth information. In Section 3, we present our hierarchical self-organizing architecture and the learning GWR algorithm extended for the classification of new action samples. In Section 4, we provide experimental results along with an evaluation of our classification algorithm on a dataset of 10 full-body actions (Parisi et al., 2014c) and a benchmark of domestic actions CAD-60 (Sung et al., 2012). We conclude in Section 5 with a discussion on the neurobiological aspects of action recognition and foundations underlying our approach, as well as future work directions for recognition systems embedded in assistive robots and HRI scenarios.

2. Recognition of Human Actions

2.1. Processing of Pose and Motion in Biology

In humans, the skill to recognize human movements arises in early life. The ability of neonates to imitate manual gestures suggests that the recognition of complex motion may depend on innate neural mechanisms (Meltzoff and Moore, 1977). Studies on preferential looking with 4-month-old infants evidence a preference for staring at human motion sequences for a longer duration than sequences with random motion (Bertenthal and Pinto, 1993). Behavioral testing has shown that young children aged three to five steadily enhance their skills to identify human and non-human biological motion portrayed as animations of point-light tokens and reach adult performance by the age of five (Pavlova et al., 2001). Psychophysiological experiments on the discrimination of actions reported a remarkable efficiency of adult human observers to temporally integrate biological motion also under noisy conditions, i.e., impoverished and potentially ambiguous visual stimuli (Neri et al., 1998). However, action perception has been shown to be disrupted by perturbations in the temporal relations of both biological and artificial motion morphs (Bertenthal and Pinto, 1993; Jastorff et al., 2006), suggesting that the recognition of complex motion is highly selective to temporal order (Giese and Poggio, 2003). Additionally, it has been found that learning plays an important role in complex motion discrimination. Studies showed that the recognition speed and accuracy of humans have improved after a number of training sessions, not only for biologically relevant motion but also for artificial motion patterns underlying a skeleton structure (Jastorff et al., 2006; Hiris et al., 2007).

Early neurophysiological studies have identified a specialized area for the visual coding of complex, articulated motion in the non-human mammalian brain (Perret et al., 1982). An extensive number of supplementary studies has shown that the mammalian visual system processes biological motion in two separate neural pathways (Giese and Poggio, 2003). The ventral pathway recognizes sequences of snapshots of body postures, while the dorsal pathway recognizes movements in terms of optic-flow patterns. Both pathways comprise hierarchies that extrapolate visual features with increasing complexity of representation. Although there has been a long-standing debate on which visual cue was predominant to action coding, i.e., either posture (Lange et al., 2006) or motion (Troje, 2002), additional studies have found neurons in the macaque superior temporal sulcus (STS) that are sensitive to both motion and posture for representing similarities among actions, thus suggesting contributions from converging cues received from the ventral and dorsal pathways (Oram and Perrett, 1996). On the basis of additional studies showing that neurons in the human STS activate by body articulation (Beauchamp et al., 2003), there is a consensus that posture and motion together play a key role in biological motion perception (Garcia and Grossman, 2008; Thirkettle et al., 2009). These findings have served to the development of architectures using learned prototype patterns to recognize actions, consistent with the idea that STS neurons integrate both body pose and motion (Vangeneugden et al., 2009). Computational feed-forward models have been developed to learn action dynamics processed as pose-motion cue patterns with recognition selective to temporal order (Giese and Poggio, 2003; Layher et al., 2012; Tan et al., 2013).

2.2. Machine Learning and Depth-Based Recognition

Other methodologies without biological foundations have also been successfully applied to action recognition. Machine learning techniques processing multi-cue features from natural images have shown motivating results for classifying a set of training actions. For instance, Xu et al. (2012) presented a system for action recognition using dynamic poses by coupling local motion information with pose in terms of skeletal joint points. They generated a codebook of dynamic poses from two RGB action benchmarks (KTH and UCF-Sports), and then classified these features with an Intersection Kernel Support Vector Machine. Jiang et al. (2012) explored a prototype-based approach using pose-motion features in combination with tree-based prototype matching via hierarchical clustering and look-up table indexing for classification. They evaluated the algorithm on the Weizmann, KTH, UCF Sports, and CMU action benchmarks. To be noted is that although these two approaches use pose-motion cues to enhance classification accuracy with respect to traditional single-cue approaches, they do not take into account an integration function that learns order-selective prototypes of joint pose-motion representations of action segments from training sequences. Furthermore, these classification algorithms can be susceptible to noise or missing observations which may occur during live recognition.

Learning systems using depth information from low-cost sensors are increasingly popular in the research community encouraged by the combination of computational efficiency and robustness to light changes in indoor environments. In recent years, a large number of applications using 3D motion information has been proposed for human activity recognition such as classification of full-body actions (Faria et al., 2014; Shan and Akella, 2014; Parisi et al., 2014c), fall detection (Rougier et al., 2011; Mastorakis and Makris, 2012; Parisi and Wermter, 2013), and recognition of hand gestures (Suarez and Murphy, 2012; Parisi et al., 2014a,b; Yanik et al., 2014). A vast number of depth-based methods has used a 3D human skeleton model to extract relevant action features for the subsequent use of a classification algorithm. For instance, Sung et al. (2012) combined the skeleton model with Histogram of Oriented Gradient features and then used a hierarchical maximum entropy Markov model to classify 12 different actions. The learning model used a Gaussian mixture model to cluster and segment the original training data into activities. Using the same action benchmark for the evaluation, Shan and Akella (2014) used action templates computed from 3D body poses to train multiple classifiers: Hidden Markov Model, Random Forests, K-Nearest Neighbor, and Support Vector Machine (SVM). Faria et al. (2014) used a dynamic Bayesian Mixture Model designed to combine multiple classifier likelihoods and compute probabilistic body motion. Zhu et al. (2014) evaluated a set of spatio-temporal interest point features from raw depth map images to classify actions with a SVM. Experiments were conducted also using interest points in combination with skeleton joint positions and color information, obtaining better results. However, the authors also showed that noisy depth data and cluttered background have a great impact on the detection of interest points, and that actions without much motion are not well recognized. The performance of the above mentioned approaches on the CAD-60 benchmark is listed in **Table 2** (Section 4).

3. Self-Organizing Neural Architecture

3.1. Overview

Our architecture consists of a two-stream hierarchy of Growing When Required (GWR) networks that processes extracted pose and motion features in parallel and subsequently integrates clustered neuronal activation trajectories from both streams. This latter network resembles the response of STS model neurons encoding sequence-selective prototypes of action segments in the joint pose-motion domain. An overall overview of the architecture is depicted in **Figure 1**. To enable the classification of new action samples, we assign labels to STS prototype neurons by extending the GWR algorithm with two offline labeling functions. We process pose and motion cues under the assumption that action recognition is selective for temporal order (Bertenthal and Pinto, 1993; Giese and Poggio, 2003). Therefore, positive recognition of action segments occurs only when neurons along the hierarchy are activated in the correct order of learned movement sequences.

3.2. Input-Driven Self-Organization

The visual system is composed of topographically arranged structures that organize according to environmental stimuli (Hubel and Wiesel, 1962; von der Malsburg, 1973; Hubel and Wiesel, 1977; Miikkulainen et al., 2005). This neural foundation, referred to as input-driven self-organization, has been shown to shape the connections in the visual cortex according to the distribution of the inputs. From a computational perspective, self-organization is an unsupervised mechanism that allows to learn representations of the input by adaptively obtaining a projection of the feature space (Kohonen, 1993).

Similar to biological mechanisms for synaptic plasticity in areas of the visual cortex, computational models may exhibit learning capabilities through the development of lateral connections between nodes governed by the principle formulated by Hebb (1949), in which nodes that are concurrently activated increase their synaptic strength. The simplest formulation of the Hebbian rule is as follows:

$$\Delta C_{ij} \propto y_i \cdot y_j \quad , \qquad (1)$$

denoting that the change of the connection strength C_{ij} is proportional to the presynaptic activity w_i and the postsynaptic activity w_j. Self-organizing networks introduce competition among nodes such that connectivity patterns become structured by activating only the neuron with the highest similarity to the input, and thus progressively reflecting topological properties of the input distribution. This mechanism, referred to as competitive Hebbian learning (CHL) (Martinetz, 1993), creates (or strengthens) the connection between the winner and the second-nearest neuron during the learning phase. The best matching neuron w_b is computed using a distance function (usually an Euclidean metric) so that, for an input signal ξ and the set of g neurons, the following condition holds:

$$\|\xi - w_b\| < \|\xi - w_g\| \quad . \qquad (2)$$

Neural network approaches inspired by biological self-organization such as self-organizing maps (SOM) (Kohonen, 1995) and neural gas (NG) (Martinetz and Schluten, 1991) have shown to be a simplified, yet plausible model for clustering human motion patterns in terms of multi-dimensional flow vectors (Parisi and Wermter, 2013). The advantage of these networks lies in their ability to learn the topological relations of the input space without supervision. The process is carried out with the use of the vector quantization technique in which a layer of competitive neurons will represent prototype vectors that encode a submanifold of the input space with a small representation error. In the SOM, each neuron of the competitive layer is connected to adjacent neurons by a neighbourhood relation that defines the structure of the map. Growing self-organizing networks represent one approach to address the limitations of the SOM and NG in which the number of neurons must be fixed beforehand and cannot be changed over time. The Growing Neural Gas (GNG) proposed by Fritzke (1995) has the ability to add new neurons to an initially small network by evaluating local statistical measures on the basis of previous adaptations, and to create and remove connections between existing neurons. The network topology is generated incrementally through CHL, i.e., for each input vector, a connection is generated between the neuron that best matches the input and the second-best matching neuron. New neurons are added when the number of learning iterations performed is a multiple of a predefined constant. This allows us to use the GNG algorithm also in on-line learning scenarios. However, the fixed

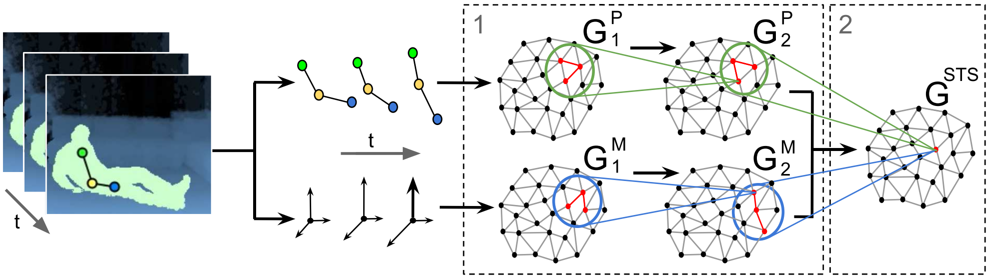


FIGURE 1 | GWR-based architecture for the processing of pose-motion samples. (1) Hierarchical processing of pose-motion features in parallel. (2) Integration of neuron trajectories in the joint pose-motion feature space.

neuron insertion interval has the limitation that the network grows at the same rate no matter how the input distribution is changing.

3.3. A Growing When Required Network

The Growing When Required (GWR) algorithm by Marsland et al. (2002) decides when to add new neurons by evaluating the activity of that neuron that best matches the current input. The GWR network is composed of a set of neurons, from now on referred to as *nodes*, with their associated weight vectors, and the edges that link the nodes. Similar to the GNG, the GWR network topology is generated through CHL (Martinetz, 1993). However, in the GWR nodes can be created at any time depending on the input. The network starts with a set of two nodes randomly initialized from within the training data. At each time step, both the nodes and the edges can be created and removed. The node activity is computed as a function of the distance between the input and the node weights. Furthermore, each node is equipped with a mechanism to measure how often the node has fired to foster the training of existing nodes over creating unnecessary ones. Edge connections have an associated age that will be used to remove old connections. At each iteration, nodes without connections are deleted.

The learning is carried out by adapting the position of the best-matching neurons and its neighbors. This learning mechanism takes into account the number of times that a node has fired so that nodes that have fired frequently are trained less. In animals, this decreasing response of neurons to a stimulus that has been frequently presented is known as habituation (Kohonen, 1993). Stanley (1976) proposed a differential equation as a simplified model of how the efficacy of an habituating synapse reduces over time:

$$\tau \frac{dh_s(t)}{dt} = \alpha [h_0 - h(t)] - S(t) \quad , \tag{3}$$

where $h_s(t)$ is the size of the firing rate for node s, h_0 is a the resting value, $S(t)$ is the stimulus strength, and τ, α are constants that control the behavior of the curve. The solution to Equation (3) can therefore provide a habituation counter $h(t)$ of how frequently a node s has fired:

$$h(t) = h_0 - \frac{S(t)}{\alpha} \cdot (1 - e^{(-\alpha_t/\tau)}) \quad . \tag{4}$$

The GWR algorithm will iterate over the training set until a given stop criterion is met, e.g., a maximum network size (number of nodes) or a maximum number of iterations.

Let A be the set of nodes, $C \subset A \times A$ the set of connections between them, $P(\xi)$ the distribution of the input ξ of dimension k, and w_n the k-dimensional weight vector of a node $n \in A$. The GWR training algorithm is given by **Algorithm 1** (Marsland et al., 2002).

The values for the reported experiments with stationary datasets were: insertion thresholds $a_T = 0.95$, learning rates $\epsilon_b = 0.2$ and $\epsilon_n = 0.006$, maximum age threshold $a_{max} = 50$, firing counter $h_0 = 1$, and habituation parameters $\alpha_b = 0.95$, $\alpha_n = 0.95$, and $\tau_b = 3.33$.

Algorithm 1 Growing When Required

```
1: Start with a set $A$ consisting of two map nodes, $n_1$ and $n_2$, at random positions.
2: Initialize an empty set of connections $C = \emptyset$.
3: At each iteration, generate an input sample $\xi$ according to the input distribution $P(\xi)$.
4: For each node $i$ calculate the distance from the input $\|\xi - w_i\|$.
5: Select the best matching node and the second-best matching node such that: $s = \arg\min_{n \in A} \|\xi - w_n\|$, $t = \arg\min_{n \in A/\{s\}} \|\xi - w_n\|$.
6: Create a connection $C = C \cup \{(s, t)\}$ if it does not exist and set its age to 0.
7: Calculate the activity of the best matching unit: $a = exp(-\|\xi - w_s\|)$.
8: If $a <$ activity threshold $a_T$ and firing counter $<$ firing threshold $f_T$ then:   Add a new node between $s$ and $t$: $A = A \cup \{(r)\}$   Create the weight vector: $w_r = 0.5 \cdot (w_s + \xi)$   Create edges and remove old edge: $C = C \cup \{(r, s), (r, t)\}$ and $C = C/\{(s, t)\}$.
9: Else, i.e., no new node is added, adapt the positions of the winning node and its neighbours $i$:   $\Delta w_s = \epsilon_b \cdot h_s \cdot (\xi - w_s)$   $\Delta w_i = \epsilon_n \cdot h_i \cdot (\xi - w_i)$   where $0 < \epsilon_n < \epsilon_b < 1$ and $h_s$ is the value of the firing counter for node $s$.
10: Increment the age of all edges connected to $s$: $age_{(s,i)} = age_{(s,i)} + 1$.
11: Reduce the firing counters according to Equation (2): $h_s(t) = h_0 - \frac{S(t)}{\alpha_b} \cdot (1 - exp(-\alpha_b t/\tau_b))$
    $h_i(t) = h_0 - \frac{S(t)}{\alpha_n} \cdot (1 - exp(-\alpha_n t/\tau_n))$.
12: Remove all edges with ages larger than $a_{max}$ and remove nodes without edges.
13: If the stop criterion is not met, go to step 3.
```

3.4. Noise Detection

The presence of noise in the sense of outliers in the training set has been shown to have a negative influence on the formation of faithful topological representations using SOMs (Parisi and Wermter, 2013), whereas such an issue is partially addressed by incremental networks. For instance, incremental networks such as GNG and GWR are equipped with a mechanism to remove rarely activated nodes and connections that may represent noisy input (**Algorithm 1**, step 12). In contrast to GNG, however, the learning strategy of the GWR shows a quick response to changes in the distribution of the input by creating new neurons to match it. The insertion threshold a_T modulates the number of neurons that will be added, e.g., for high values of a_T more nodes will be created (**Algorithm 1**, step 8). However, the network is also equipped with a mechanism to avoid slight input fluctuations to perturb the learning convergence and the creation of unnecessary nodes. The GWR takes into account the number of times that a neurons has been activated, so that neurons that have been activated more times, are trained less. Therefore, an additional threshold modulates the firing counter of neurons so that during the learning process less trained neurons are updated, whereas new neurons are created only when existing neurons do not sufficiently represent the input. A number of experiments have shown that the GWR is well-suited for novelty detection (Marsland et al., 2002), which involve the identification of inputs that do not fit the learned model.

In line with this mechanism, we use the activation function (**Algorithm 1**, step 7) to detect noisy input after the training phase. The activation function will be equal to 1 in response to input that perfectly matches the model, i.e., minimum distance between the weights of the neuron and the input, and will decrease exponentially for input with a higher distance. If the response of the network to the novel input is below a given novel activation threshold a_{new}, then the novel input can be considered noisy in the sense that it is not represented by well-trained prototype neurons, and thus discarded. The threshold value a_{new} can be empirically selected by taking into account the response distribution of the trained network with respect to the training set. For each novel input x_{new}, we compute:

$$exp\left\{-\sqrt{\sum_{j=1}^{k}(x_{new,j} - s(x_{new,j}))^2}\right\} < \bar{A} - \gamma \cdot \sigma(A) \quad , \qquad (5)$$

where $\bar{A}$ and $\sigma(A)$ are respectively the mean and the standard deviation of the set of activations A is obtained from the training set, and γ is a constant value that modulates the influence of fluctuations in the activation distribution. **Figure 2** shows a GWR network trained with 100 input vectors with two normally distributed clusters. Over its 500 iterations, the network created 556 neurons and 1145 connections ($a_T = 0.95, \gamma = 4$). The activation values for a test set of 200 samples (also normally distributed) containing artificially introduced noise are shown in **Figure 3**. It is observable how noisy samples lie below the computed activation threshold $a_{new} = 0.1969$ (Equation 5) and can, therefore, be discarded. We use this noise detection procedure to all the networks in our architecture with the aim to attenuate noise in the training data and prevent the forced classification of input that are not represented by the trained model.

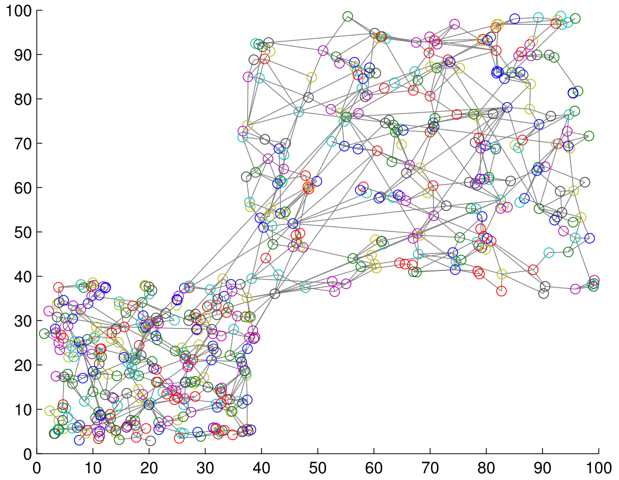


FIGURE 2 | A GWR network trained with a normally distributed training set of 1000 samples resulting in 556 nodes and 1145 connections.

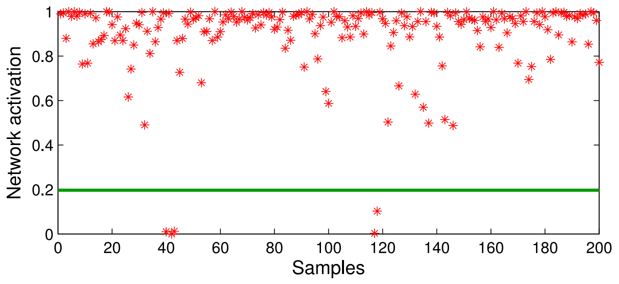


FIGURE 3 | Activation values for the network trained in Figure 2 with a test set of 200 samples containing noise. Noisy samples line under novelty threshold $a_{new} = 0.1969$ (green line).

3.5. Hierarchical Learning and Integration

The motivation underlying our hierarchical learning is to use trajectories of neuron activations from one network as input for the training for a subsequent network. This mechanism allows to obtain progressively specialized neurons coding inherent spatio-temporal dependencies of the input, consistent with the assumption that the recognition must be selective for temporal order.

Hierarchical training is carried out as follows. We first train a network G with a training set T. After the training is completed, the subsequent network G^* will be trained with a new set T^* that is obtained computing trajectories of best-matching neurons from G for samples of T. For each k-dimensional sample $x \in T$, we compute the best-matching neuron as

$$s(x) = \arg\min_{n \in A} \sqrt{\sum_{j=1}^{k}(x_j - w_{n,j})^2} \quad , \qquad (6)$$

from which we can compute a trajectory of prototype neurons of length q:

$$\omega(x_i) = \{s(x_i), s(x_{i-1}), \ldots, s(x_{i-q+1}), i \in [q..m]\} \quad , \qquad (7)$$

where m is the number of samples of T. The next step is to compute the training set T^* by concatenating the $m - q$ trajectories of neuron activations over T with a temporal sliding window scheme, in our specific case using activation trajectories with 3 neurons ($q = 3$) for all the stages. The training of G^* will then produce a network with neurons encoding temporally-ordered prototype sequences from consecutive samples of T.

At the first stage of our hierarchy, each stream is composed of two GWR networks to process pose and motion features separately. We therefore compute two distinct datasets with sequentially-ordered pose and motion features, denoted as P and M respectively. Since P and M are processed by different network hierarchies, they can differ in dimensionality. Following the notation introduced in **Figure 1**, we train the networks G_1^P and G_1^M with samples from P and M respectively. After this step, we train G_2^P and G_2^M with the training sets of concatenated trajectories of best-matching neurons (Equation 7).

The STS stage consists of the integration of prototype activation trajectories from both streams by training the network G^{STS} with two-cue trajectory samples. For this purpose, we compute a new dataset T^{STS} by merging best-matching trajectories from G_2^P and G_2^M into a set of trajectory pairs ψ_u as follows:

$$\psi_u = \{s(\omega(x_i)), \ldots, s(\omega(x_{i-q-1})), s(\omega(y_i)), \ldots, s(\omega(y_{i-q-1})), x_i \in P, y_i \in M, u \in [q..m-q]\} \quad . \tag{8}$$

After the training of G^{STS} is completed, each neuron will encode a sequence-selective prototype action segment, thereby integrating changes in the configuration of a person's body pose over time.

3.6. Classification

At recognition time, our goal is to process and classify unseen action sequences to match one of the training actions. For this purpose, we extend the unsupervised GWR-based learning with two labeling functions: one for the training phase and one for returning the label of unseen samples.

Let L be the set of j action classes that we want to recognize, for instance "walk" and "fall down." We then assume that each action δ_j will be therefore composed of a set of labeled, sequentially-ordered feature vectors:

$$\delta_j = \{(F_i, l_j) : i \in [1..v], l_j \in L\} \quad , \tag{9}$$

where l_j is the action label and v is the number of feature vectors $f \in F_i$ for the action class δ_j. Sample labels are not used during the first stage of the learning process. The learning process is carried out without supervision following **Algorithm 1**. In addition, each neuron of the STS network will be assigned an action label during the training phase. We train the G^{STS} network with the labeled training pairs (ψ_u, l_j) and define a labeling function $l : N \rightarrow L$ for the training phase, where N is the set of nodes. We adopted the labeling technique that has shown to achieve best classification accuracy among other labeling strategies for GNG-based learning discussed by Beyer and Cimiano (2011). According to a minimal-distance strategy, the sample ψ_u will adopt the label l_j of the closest ψ:

$$l(\psi_k) = l_j = l(\arg\min_{\psi \in \Psi} \|\psi_i - \psi\|^2) \quad . \tag{10}$$

This labeling procedure works in an offline mode since we assume that all training samples and labels are available a priori. This mechanism requires the extension of the standard GWR algorithm for assigning a training label to the best-matching neuron of the current input (**Algorithm 1**, step 5).

For the classification task, we define a recognition function $\varphi : \Psi \rightarrow L$ on the basis of a single-linkage strategy (Beyer and Cimiano, 2011) in which a new sample Ψ_{new} is labeled with l_j associated to the neuron n that minimizes the distance to the new sample:

$$\varphi(\psi_{new}) = \arg\min_{l_j}(\arg\min_{n \in N(l_j)} \|n - \psi_{new}\|^2) \quad . \tag{11}$$

The hierarchical flow is composed of 3 networks with each subsequent network neuron encoding a window of 3 samples from the previous one. Therefore, this classification algorithm returns the first action label l_{new} after 9 new samples $\hat{f} \in F$. Then, applying the temporal sliding window scheme, we get a new action label for each new sample. For instance, operating at 15 frames per second, we would get the first action label after $9/15 = 0.6$ s.

4. Results

We evaluated our approach both on our action dataset (Parisi et al., 2014c) and the public action benchmark CAD-60 (Sung et al., 2012). We now provide details on feature extraction, learning parameters for the GWR-based training and recognition, and a comparative evaluation.

4.1. Action Features

4.1.1. Full-body actions

Our action dataset is composed of 10 full-body actions performed by 13 student participants with a normal physical condition. Participants were naive as to the purpose of the experiment and they had not been explained how to perform the actions in order to avoid biased execution. They were recorded individually and gave written consent to participate in the study. We monitored the participants in a home-like environment with a Kinect sensor installed 130 m above the ground. Depth maps were sampled with a VGA resolution of 640 × 480, an operation range from 0.8 to 3.5 meters and a constant frame rate of 30 Hz. The dataset contained periodic and goal-oriented actions:

- Periodic: Standing, walking, jogging, sitting, lying down, crawling (10 min each);
- Goal-oriented: Pick up object, jump, fall down, stand up (60 repetitions each).

From the raw depth map sequences, 3D body joints were estimated on the basis of the tracking skeleton model provided by OpenNI[1]. We represented whole-body actions in terms of three body centroids (**Figure 4**): C_1 for upper body with respect to the shoulders and the torso; C_2 for middle body with respect to the torso and the hips; and C_3 for lower body with respect to the hips and the knees. Each centroid is computed as a point sequence of real-world coordinates $C = (x, y, z)$. To attenuate sensor noise, we used the median value of the last 3 estimated points. We then estimated upper and lower orientations θ^u and θ^l given by the slope angles of the line segments $\{C_1, C_2\}$ and $\{C_2, C_3\}$ respectively. As shown in **Figure 4**, the values θ^u and θ^l describe the overall body pose according to the orientation of the torso and the legs, which allows to capture significant pose configurations in actions such as walking, sitting, picking up and lying down. We computed the body velocity S_i as the difference in pixels of the centroid C_1 between two consecutive frames i and $i - 1$. The upper centroid was selected based on the motivation that the orientation of the torso is the most characteristic reference during the execution of a full-body action (Papadopoulos et al.,

[1] OpenNI SDK. http://openni.ru/openni-sdk/.

2014). We then computed horizontal speed h_i and vertical speed v_i (Parisi and Wermter, 2013). For each action frame i, we computed the following pose-motion vector:

$$F_i = (\theta_i^u, \theta_i^l, h_i, v_i) \quad . \tag{12}$$

Thus, each action A_j will be composed of a set of sequentially ordered pose-motion vectors such that:

$$A_j := \{(F_i, l_j) : i \in [1..n], l_j \in L\} \quad , \tag{13}$$

where l_j is the action label, L is the set of class labels, and n is the number of training vectors for the action j. Action labels were manually annotated for video sequences containing one action. We divided the data equally into training and test set, i.e., 30 sequences of 10 s for each periodic action and 30 repetitions for each goal-oriented action. Both the training and test sets contained data from all participants. For a fair comparison with previous results (Parisi et al., 2014c), we adopted similar feature extraction and evaluation schemes.

4.1.2. CAD60

The Cornell activity dataset CAD-60 (Sung et al., 2012) is composed of 60 RGB-D videos of four subjects (two males, two females, one left-handed) performing 12 activities: *rinsing mouth, brushing teeth, wearing contact lens, talking on the phone, drinking water, opening pill container, cooking (chopping), cooking (stirring), talking on couch, relaxing on couch, writing on whiteboard, working on computer*. The activities were performed in 5 different environments: office, kitchen, bedroom, bathroom, and living room. The videos were collected with a Kinect sensor with distance ranges from 1.2 to 3.5 m and a depth resolution of 640×480 at 15 frames per second. The dataset provides raw depth maps and RGB images, and skeleton data. An example of the actions and the resulting skeletons is shown in **Figure 5**. The dataset provides skeleton data composed of 15 extracted joints for the following body parts: *head, neck, torso, shoulders, elbows, hands, hips, knees,* and *feet*.

For our approach, we used the set of 3D positions without the *feet*, leading to 13 joints (i.e., 39 input dimensions). Instead of using world coordinates, we encoded the joint positions using the center of the hips as frame of reference to obtain translation invariance. We then computed joint motion as the difference of two consecutive frames for each pose transition. We added a mirrored version of all action samples to obtain invariance to actions performed with either the right or the left hand.

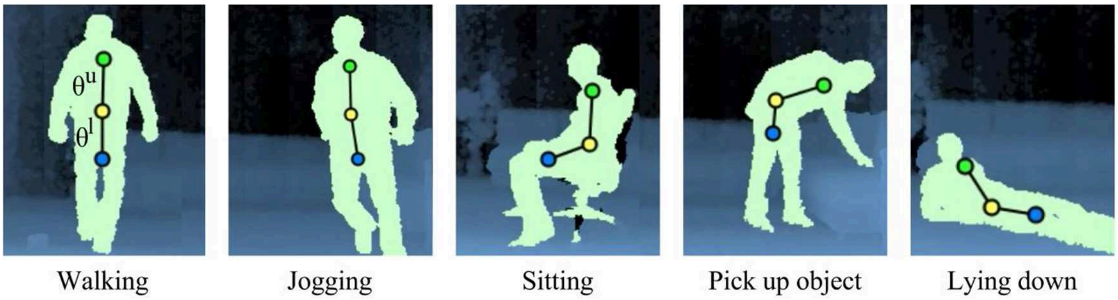


FIGURE 4 | Representation of full-body movements from our action dataset. We estimate three centroids C_1 (green), C_2 (yellow) and C_3 (blue) for upper, middle and lower body respectively. The segment slopes θ^u and θ^l describe the posture in terms of the overall orientation of the upper and lower body.

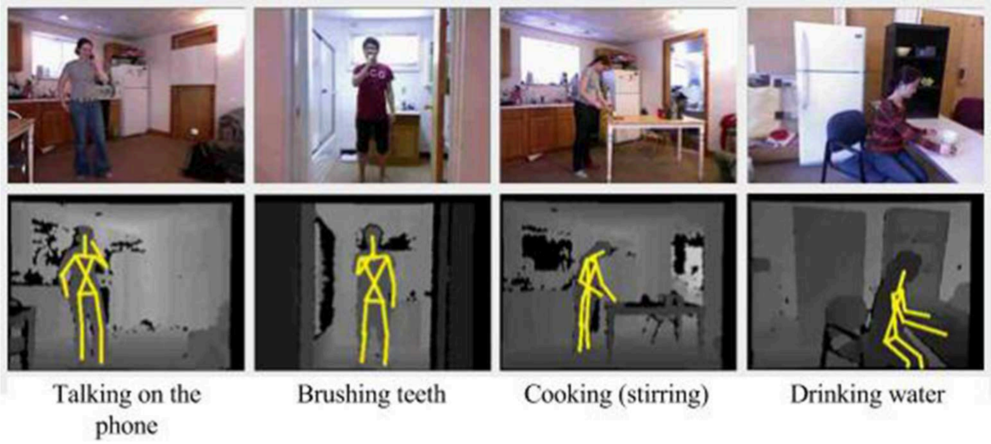


FIGURE 5 | Daily actions from the CAD-60 dataset (RGB and depth images with skeleton).

4.2. Training

We now report the GWR parameters for the training sessions. We set the following values: insertion thresholds $a_T = 0.90$, learning rates $\epsilon_b = 0.3$, and $\epsilon_n = 0.006$, maximum age $a_{max} = 50$, firing counter parameters $h_0 = 1$, $\tau_b = 0.3$, $\tau_n = 0.1$. Each network stopped training after a 500 epochs over the whole dataset. These parameters were empirically found to let the model learn spatio-temporal dependencies with the best accuracy in terms of classification labels returned by the last network G^{STS}. For a single network, the number of neurons converged already after 100 epochs, and weight vectors of neurons showed little modification after 400 epochs. If we consider the 2 networks per stream in the first stage of the hierarchy and the integration network in the second stage (**Figure 1**), it took overall 1500 epochs to obtained a trained neuron in the G^{STS} network.

In **Table 1**, we show the resulting properties of the networks along the hierarchy after the training sessions on the two datasets. In both cases, it can be observed that the number of nodes (N) and connections (C) is lower for higher levels of the hierarchy. The lower numbers indicate that in the STS level neurons encode more complex spatio-temporal dependencies with respect to the first level (in which only uni-cue spatial relations are considered), but with a smaller number of specialized neurons. To be noticed is that the number of neurons did not depend on the dimensionality of the input, but rather on the distribution of the data. From **Table 1** it can also be seen that the activation threshold (a) increases toward higher levels of the hierarchy. In the first level, the activation function yielded larger fluctuations due to outliers and input data that were rarely presented to the network during the training. Conversely, activations of training samples matching the model get higher as neurons specialize. These results indicate that noise from the training data was not propagated along the hierarchy, but rather detected and discarded, which leads to a larger a-value.

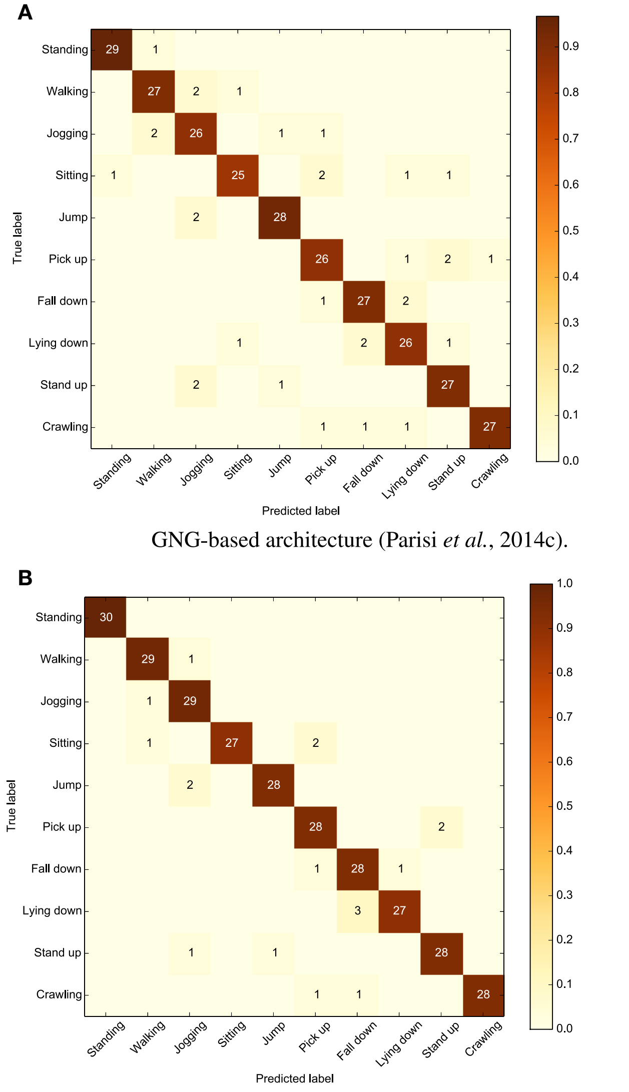
4.3. Evaluation

4.3.1. Full-Body Actions

Similar to previously reported results (Parisi et al., 2014c), we evaluated the system on 30 sequences of 10s for each periodic action and 30 repetitions for each goal-oriented action. Experiments showed that our new approach outperforms the previous one with an average accuracy rate of 94% (5% higher the than GNG-based architecture using an extra network for noise detection, and 18% higher than the same architecture without noise detection). We show the confusion matrix for both the approaches in **Figure 6** (with each row of the matrix being an instance of the actual actions and each column an instance of the predicted actions). We can observe from the matrices that all the actions are slightly classified more accurately with respect to Parisi et al. (2014c). The most misclassified actions are "sitting" and "laying down." In the first case, the action was confused with "walking" and "pick up." This misclassification was mostly caused by skeleton tracking errors, i.e., when sitting down, the self-occlusion of joints may compromise the

TABLE 1 | Training results on the two datasets—For each trained network along the hierarchy, the table shows the resulting number of nodes (*N*) and connections (*C*), and the activation threshold (*a*).

Full-body actions	G_1^P	$N = 225$ $C = 435$ $a = 0.1865$	G_2^P	$N = 183$ $C = 338$ $a = 0.1934$	G^{STS}	$N = 118$ $C = 378$ $a = 0.2932$
	G_1^M	$N = 254$ $C = 551$ $a = 0.1732$	G_2^M	$N = 192$ $C = 353$ $a = 0.1910$		
CAD-60	G_1^P	$N = 289$ $C = 403$ $a = 0.1778$	G_2^P	$N = 214$ $C = 445$ $a = 0.1898$	G^{STS}	$N = 137$ $C = 309$ $a = 0.2831$
	G_1^M	$N = 302$ $C = 542$ $a = 0.1698$	G_2^M	$N = 239$ $C = 495$ $a = 0.1991$		



GNG-based architecture (Parisi *et al.*, 2014c).

GWR-based architecture with embedded noise detection.

FIGURE 6 | Confusion matrices for our dataset of 10 actions showing better results for our GWR-based architecture (average accuracy 94%) compared to our previous GNG-based approach (89%).

estimation of the overall body pose. The action "laying down" was, instead, misclassified as "fall down." This is likely caused by the horizontal body poses shared between the two actions, despite the contribution of motion to disambiguate actions with similar poses.

4.3.2. CAD60

For our evaluation on the CAD-60 dataset, we adopted a similar scheme as the one reported by Sung et al. (2012) using all the 12 activities plus a random action with *new person* strategy, i.e., the first 3 subjects for training and the remaining for test purposes. In **Table 3**, we show a comparison of our results with the state of the art on the CAD-60 dataset with precision and recall as evaluation metrics, and ranked by the F_1-score computed as:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad . \tag{14}$$

We obtained 91.9% precision, 90.2% recall, and 91% F-score, indicating that our model exhibits a good positive predictive value and very satisfactory sensitivity to classified actions. Precision and recall for each action and environment are shown in **Table 3**. To be noted is that we separated the actions into 5 different environments for a consistent and more informative comparison with other approaches using the same dataset, whereas the specific properties of the environments were not known to the model and had no effect on the segmentation of the skeleton joints, therefore not influencing the classification process.

The best state-of-the-art result has 93.8% precision, 94.5% recall, and 94.1% F-score (Shan and Akella, 2014). In their work, the authors identified a number of key poses prior to learning from which they compute spatio-temporal action templates, which makes this approach highly data-dependent. Each action must be segmented into atomic action templates composed of a set of n key poses, where n depends on the action's duration and complexity. Furthermore, experiments with low-latency (close to real-time) classification have not been reported. The second approach with slightly better results than ours is the work by Faria et al. (2014) with 93.2% precision, 91.9% recall, and 91.5% F-score. In their work, the authors used a dynamic Bayesian Mixture Model to classify motion relations between body poses. However, they used the raw depth images to estimate their own skeleton model (and did not use the one provided by the CAD-60 benchmark dataset). Therefore, differences in the tracked skeleton may exist that hinder a quantitative comparison with our classification method.

TABLE 2 | Precision and recall of our approach evaluated on the 12 activities from in CAD60 and comparison with other algorithms.

Algorithm	Precision (%)	Recall (%)	*F*-score (%)
Sung et al., 2012	67.9	55.5	61.1
Ni et al., 2013	75.9	69.5	72.1
Koppula et al., 2013	80.8	71.4	75.8
Gupta et al., 2013	78.1	75.4	76.7
Gaglio et al., 2014	77.3	76.7	77
Zhang and Tian, 2012	86	84	85
Zhu et al., 2014	93.2	84.6	88.7
Our approach	**91.9**	**90.2**	**91**
Faria et al., 2014	91.1	91.9	91.5
Shan and Akella, 2014	93.8	94.5	94.1

Bold values indicate the classification results for our algorithm.

TABLE 3 | Precision, recall, and *F*-score of our approach on the five environments of the CAD-60 dataset.

Location	Activity	Precision (%)	Recall (%)	*F*-score (%)
Office	Talking on the phone	94.1	92.8	93.4
	Drinking water	92.9	91.5	92.2
	Working on computer	94.3	93.9	94.1
	Writing on whiteboard	95.7	94.0	94.8
	Average	94.3	93.1	93.7
Kitchen	Drinking water	93.2	91.4	92.3
	Cooking (chopping)	86.4	86.7	86.5
	Cooking (stirring)	88.2	86.2	87.2
	Opening pill container	90.8	84.6	87.6
	Average	89.7	87.2	88.4
Bedroom	Talking on the phone	93.7	91.9	92.8
	Drinking water	90.9	90.3	90.6
	Opening pill container	90.8	90.1	90.4
	Average	91.8	91.7	91.7
Bathroom	Wearing contact lens	91.2	87.0	89.1
	Brushing teeth	90.6	88.0	89.3
	Rinsing mouth	87.9	85.8	86.8
	Average	89.9	86.9	88.4
Living room	Talking on the phone	94.8	92.1	93.4
	Drinking water	91.7	90.8	91.2
	Relaxing on couch	93.9	91.7	92.8
	Talking on couch	94.7	93.2	93.9
	Average	93.8	92.0	92.9

5. Discussion

5.1. Summary

In this paper, we presented a neurobiologically-motivated architecture that learns to recognize actions from depth map video sequences. The proposed approach relies on three assumptions that are consistent with evidence on neural mechanisms for action discrimination: (1) pose and motion action features are processed in two distinct pathways, respectively the ventral and the dorsal stream, and then action cues are integrated to provide a joint percept (Perret et al., 1982; Vangeneugden et al., 2009); (2) hierarchies within each pathway process features with increasing complexity (Giese and Poggio, 2003); and (3) visual information is arranged according to input-driven self-organization (von der Malsburg, 1973; Kohonen, 1993; Miikkulainen et al., 2005). Our neural architecture consists of a two-pathway hierarchy of GWR networks that process pose-motion features in parallel and subsequently integrate action cues to provide movement dynamics in the joint feature space. Hierarchical learning was carried out using prototype trajectories composed of neuron activation patterns. The learning mechanism of the network allows to attenuate noise and detect noisy novel samples during on-line classification. For classification purposes, we extended the GWR implementation with two labeling functions. The evaluation of our approach has shown that our architecture outperforms previous results on action recognition for a dataset of 10 full-body actions, and that we achieved comparable results with the state of the art for a publicly available action benchmark.

Features of action sequences were extracted from depth map videos. The use of depth sensors has received increasing attention by action recognition researchers along with the integration of such technology with mobile robot platforms and humanoids (e.g., see Fanello et al., 2013; Parisi and Wermter, 2015). This is due to the fact that devices such as Kinect and Xtion represent low-cost sensors for the efficient segmentation of human motion robust to light changes in indoor environments. These factors play an important role in the development of a robust artificial system for the recognition of actions in real-world scenarios, e.g., detection of fall events in a domestic environment (Rougier et al., 2011; Mastorakis and Makris, 2012; Parisi and Wermter, 2015). Previous research has shown that the movement of a depth sensor, e.g., when mounted on a mobile robot, introduces a greater number of noisy observations that may impair the effective detection of action events (Parisi and Wermter, 2013). Therefore, artificial systems operating in natural environments should address the tolerance of noise to cope with sensor errors and occluded persons. The use of a self-organizing GWR allows to learn an incremental number of training actions and embed the mitigation of noisy samples into the learning mechanism. With this scheme, outliers in the training set do not propagate along the hierarchy during the training, and can automatically be detected during live classification (further details are discussed in Section 5.2).

5.2. Analogies with Biological Findings

The GWR networks (Marsland et al., 2002) have the ability to dynamically change their topological structure through competitive Hebbian learning (Martinetz, 1993) to incrementally match the distribution of the data in input space, thereby mimicking input-driven synaptic plasticity (Pascual-Leone et al., 2011) exhibited by some areas of the visual cortex (Hubel and Wiesel, 1962, 1977; Miikkulainen et al., 2005). Furthermore, this learning mechanism creates new neurons taking into account how well trained existing neurons are. This is achieved through a simplified model of the habituation process (Thompson and Spencer, 1966) and the benefits are twofold. First, it allows the convergence of the network in the sense that well-trained neurons will stop being updated. Second, the network responds quickly to changes in the distribution of the input. In this context, the insertion threshold has a strong influence on the number of neurons that will be created to match dynamic input fluctuations.

In our implementation of the GWR algorithm, we used the Euclidean distance as a metric to compute the distance of prototype neurons and neuron trajectories from the current input. Giese et al. (2008) investigated perceptual representations of full-body motion finding motion patterns that reside in perceptual spaces with well-defined metric properties. They conducted experiments with 2D and 3D joints of prototype trajectories with results implying that perceptual representations of complex motion patterns closely reflect the metric of movements in the physical world. Although more precise neural mechanisms that implement distance computation remain to be explored, we can therefore assume that the Euclidean distance is an adequate metric to compare articulated movement patterns.

For the processing of actions, we rely on the extraction of a simplified 3D skeleton model from which we estimate significant action properties, such as pose and motion, while maintaining a low-dimensional feature space. The skeleton model estimated by OpenNI, although not anatomically faithful, provides a convenient representation from which it is possible to extrapolate actor-independent action dynamics. The use of such models is in line with biological evidence demonstrating that human observers are very proficient at recognizing and learning complex motion underlying a skeleton structure (Jastorff et al., 2006; Hiris et al., 2007). These studies show that the presence of a holistic structure improves the learning speed and accuracy of action patterns, also for non-biologically relevant motion such as artificial complex motion patterns. This model may be susceptible to sensor noise and situations of partial occlusion and self-occlusion (e.g., caused by body rotation) for which body joint values may be noisy or missing. Although it may be desirable to implement invariance transformations (e.g., Sofatzis et al., 2014) or remove sensor noise (Parisi and Wermter, 2013), these limitations are not in contrast with biological evidence demonstrating that the recognition of complex motion is strongly view-dependent. Psychophysical studies showed that action recognition is impaired by biological motion stimuli being upside-down or rotated with respect to the image plane (Sumi, 1984; Pavlova and Sokolov, 2000). Furthermore, it has been found that learned visual representations seem to be highly orientation-dependent, i.e., discrimination performance increased only when the test patterns presented the same orientation as in the training (Jastorff et al., 2006). Therefore, view-dependence in recognition of complex motion is consistent with the idea that recognition is based on the matching of learned two-dimensional patterns, whereas view-independence may be achieved by means of 3D internal models (Hogg, 1983).

Our recognition scheme for action sequences is in line with a number of studies demonstrating that action discrimination is selective to temporal order (Bertenthal and Pinto, 1993; Giese and Poggio, 2003; Jastorff et al., 2006). Therefore, this task may involve learning mechanisms able to extrapolate spatio-temporal dependencies of sequences. Recurrent versions of self-organizing networks have been extensively investigated that extend the feed-forward learning mechanism with context neurons for referring to past activations, thereby allowing the processing of sequences and structured data (e.g., see Strickert and Hammer, 2005 for a recursive SOM model). Although the original implementation of the GWR processes real-valued vectors only in the spatial domain, it may be easily extended for processing sequences in a similar fashion. For instance, Andreakis et al. (2009) devised a recursive GNG network with a context layer to learn spatio-temporal patterns. However, consistently with evidence of a hierarchical architecture of the visual system (Giese and Poggio, 2003), we opted for a feed-forward architecture that exhibits progressively time-selective levels of representations. In this setting, action recognition is modulated by temporal order resulting from lateral connections that form activation trajectories between prototype neurons. Trajectories were generated with serialized concatenations of a fixed number of samples in a temporal sliding window fashion, in our specific case empirically set to trajectories of 3 neuron activations for each visual cue. This scheme is in accordance with neurophysiological evidence that actions are represented by sequences of integrated poses over fixed windows of around 120 ms (Singer et al., 2010). A series of well-established computational models have been proposed that implement a feed-forward architecture for processing action features with increasing complexity (Giese and Poggio, 2003; Lange et al., 2006; Tan et al., 2013).

5.3. Future Work

In this work, we focused on a feed-forward mechanism for learning human actions represented with pose-motion features. However, a number of studies have demonstrated that biological motion recognition is also strongly modulated by higher level cognitive representations, such as top-down influences (Bülthoff et al., 1998; Thornton et al., 2002), and representations of biomechanically plausible motion (Shiffrar and Freyd, 1990). These aspects were not considered in this paper and are part of future work.

An additional future work direction is to investigate the interplay of pose-motion cues and recognition strategies when one of the two stimuli is suppressed. At its current state, our system requires that both the pose and motion samples are available for parallel processing and integration. However, studies have shown that observers can shift between pose and motion-based strategies, depending on the available cue (Tyler et al., 2011). In other words, suppressing one of the cues does not fully impair action perception. In line with this assumption, we could extend our neural architecture with interlateral connections so that neurons from distinct pathways can co-activate in the presence of single-cue input. With our implementation, this mechanism would require neurons in GWR to be equipped with symmetric, inter-network references that link prototype neurons between the G^P and G^M populations, and enable the computing of activation trajectories in both pathways when only neurons from one pathway are activated. In this setting, the dynamics of learning and cue integration are to be investigated.

Finally, the reported results motivate the embedding of our learning system into mobile robot platforms to conduct further evaluations in more complex scenarios, where the robust recognition of actions plays a key role. For instance, the visual detection of dangerous events for assistive robotics such as fall events (Parisi and Wermter, 2013, 2015), and the recognition of actions with learning robots in HRI scenarios (Soltoggio et al., 2013a,b; Barros et al., 2014; Parisi et al., 2014a,b).

Acknowledgments

This work was jointly supported by the DAAD German Academic Exchange - Cognitive Assistive Systems Project and by the DFG German Research Foundation (grant #1247) - International Research Training Group CINACS (Cross-modal Interaction in Natural and Artificial Cognitive Systems).

References

Andreakis, N., Hoyningen-Huene, N. v., and Beetz, M. (2009). "Incremental unsupervised time series analysis using merge growing neural gas," in *Advances in Self-Organizing Maps*, eds J. C. Principe and R. Miikkulainen (St. Augustine, FL: Springer), 10–18.

Bülthoff, I., Bülthoff, H., and Sinha, P. (1998). Top-down influences on stereoscopic depth perception. *Nat. Neurosci.* 1, 254–257. doi: 10.1038/699

Barros, P., Parisi, G. I., Jirak D., and Wermter, S. (2014). "Real-time gesture recognition using a humanoid robot with a deep neural Architecture," in *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (Madrid: IEEE-RAS), 83–88.

Beauchamp, M. S., Lee, K. E., Haxby, J. V., and Martin, A. (2003) FMRI responses to video and point-light displays of moving humans and manipulable objects. *J. Cogn. Neurosci.* 15, 991–1001. doi: 10.1162/089892903770007380

Bertenthal, B. I., and Pinto, J. (1993). "Complementary processes in the perception and production of human movement," in *Dynamic Approaches to Development 2*, eds E. Thelen, and L. Smith (Cambridge: MIT Press), 209–239.

Beyer, O., and Cimiano, P. (2011) "Online labelling strategies for growing neural gas," in *IDEAL11*, eds H. Yin, W. Wang, and V. Rayward-Smith (Norwich: Springer Berlin Heidelberg), 76–83.

Blake, R., and Shiffrar, M. (2007). Perception of human motion. *Annu. Rev. Psychol.* 58, 47–73. doi: 10.1146/annurev.psych.57.102904.190152

Ellis, C., Masood, S. Z., Tappen, M. F., LaViola, J. J. Jr., and Sukthankar, R. (2013). Exploring the trade-off between accuracy and observational latency in action recognition. *Int. J. Comput. Vis.* 101, 420–436. doi: 10.1007/s11263-012-0550-7

Fanello, S. R., Gori, I., Metta, G., and Odone, F. (2013) Keep it simple and sparse: real-time action recognition. *J. Mach. Learn. Res.* 14, 2617–2640.

Faria, D. R., Premebida, C., and Nunes, U. (2014). "A probabilistic approach for human everyday activities recognition using body motion from RGB-D images," in *2014 RO-MAN: the 23rd IEEE International Symposium Robot and Human Interactive Communication* (Edinburgh: IEEE).

Fritzke, B. (1995). "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems*, Vol. 7, eds G. Tesauro, D. S. Touretzky, and T. K. Leen (Cambridge, MA: MIT Press), 625–632.

Gaglio, S., Lo Re, M., and Morana, M. (2014) Human activity recognition process using 3-D posture data. *IEEE Trans. Hum. Mach. Syst.* 99, 1–12. doi: 10.1109/THMS.2014.2377111

Garcia, J. O., and Grossman, E. D. (2008). Necessary but not sufficient: motion perception is required for perceiving biological motion. *Vis. Res.* 48, 1144–1149. doi: 10.1016/j.visres.2008.01.027

Giese, M. A., and Poggio, T. (2003). Neural mechanisms for the recognition of biological movements. *Nat. Rev. Neurosci.* 4, 179–192. doi: 10.1038/nrn1057

Giese, M. A., Thornton, I., and Edelman, S. (2008). Metrics of the perception of body movement. *J. Vis.* 8, 1–18. doi: 10.1167/8.9.13

Gupta, R., Chia, A. Y.-S., and Rajan, D. (2013). "Human activities recognition using depth images," in *ACM International Conference on Multimedia* (New York, NY: ACM), 283–292.

Han, J., Shao, L., Xu, D., and Shotton, J. (2013) Enhanced computer vision with microsoft kinect sensor: a review. *IEEE Trans. Cybern.* 43, 1318–1334. doi: 10.1109/TCYB.2013.2265378

Hebb, D. O. (1949). *The Organization of Behavior*. New York, NY: Wiley.

Hiris, E. (2007) Detection of biological and nonbiological motion. *J. Vis.* 7, 1–16. doi: 10.1167/7.12.4

Hogg, D. (1983). Model-based vision: a program to see a walking person. *Image Vis. Comp.* 1, 5–19. doi: 10.1016/0262-8856(83)90003-3

Hubel, D. H., and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.* 160, 106–154. doi: 10.1113/jphysiol.1962.sp006837

Hubel, D. H., and Wiesel, T. N. (1977). Ferrier lecture. Functional architecture of macaque monkey visual cortex. *Proc. R. Soc. Lond. B Biol. Sci.* 198, 1–59. doi: 10.1098/rspb.1977.0085

Jastorff, J., Kourtzi, Z., and Giese, M. A. (2006). Learning to discriminate complex movements: biological versus artificial trajectories. *J. Vis.* 6, 791–804. doi: 10.1167/6.8.3

Jiang, Z., Lin, Z., and Davis, L. S. (2012). Recognizing human actions by learning and matching shape-motion prototype trees. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 533–547. doi: 10.1109/TPAMI.2011.147

Kachouie, R., Sedighadeli, S., Khosla, R., and Chu, M-T. (2014). Socially assistive robots in elderly care: a mixed-method systematic literature review. *Int. J. Hum. Comput. Interact.* 30, 369–393. doi: 10.1080/10447318.2013.873278

Kohonen, T. (1993). *Self-Organization and Associative Memory, 3rd Edn.* Berlin: Springer.

Kohonen, T. (1995). "Self-organizing maps," in *Series in Information Science 30*, eds T. S. Huang, T. Kohonen, and M. R. Schroeder (Heidelberg: Springer), 502.

Koppula, H. S., Gupta, R., and Saxena, A. (2013). Learning human activities and object affordances from RGB-D videos. *Int. J. Robot. Res.* 32, 951–970. doi: 10.1177/0278364913478446

Lange, J., and Lappe, M. (2006). A model of biological motion perception from configural form cues. *J. Neurosci.* 26, 2894–2906. doi: 10.1523/JNEUROSCI.4915-05.2006

Layher, G., Giese M. A., and Neumann, H. (2012). "Learning representations for animated motion sequence and implied motion recognition," in *ICANN12* (Helderberg: Springer), 427–434.

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

Martinetz, T., and Schluten, K. (1991). "A "neural-gas" network learns topologies," in *Artificial Neural Networks*, eds T. Kohonen, K. Makisara, O. Simula, and J. Kangas (North Holland: Elsevier Science Publishers B.V.), 397–402.

Martinetz, T. (1993). "Competitive Hebbian learning rule forms perfectly topology preserving maps," in *ICANN93* (Helderberg: Springer), 427–434.

Mastorakis, G., and Makris, D. (2012). Fall detection system using Kinect's infrared sensor. *J. Real Time Image Process* 9, 635–646.

Meltzoff, A. N., and Moore, M. K. (1977). Imitation of facial and manual gestures by human neonates. *Science* 198, 74–78. doi: 10.1126/science.897687

Miikkulainen, R., Bednar, J. A., Choe, Y., and Sirosh J. (2005) *Computational Maps in the Visual Cortex.* New York ,NY: Springer.

Neri, P., Concetta Morrone, M., and Burr, D. C. (1998). Seeing biological motion. *Nature* 395, 894–896. doi: 10.1038/27661

Ni, B., Pei, Y., Moulin, P., and Yan, S. (2013). Multilevel depth and image fusion for human activity detection. *IEEE Trans. Cybern.* 43, 1383–1394. doi: 10.1109/TCYB.2013.2276433

Oram, M. W., and Perrett, D. I. (1996). Integration of form and motion in the anterior superior temporal polysensory area (STPa) of the macaque monkey. *J. Neurophysiol.* 76, 109–129.

Papadopoulos, G., Th. Axenopoulos, A., and Daras, P. (2014). "Real-time skeleton-tracking-based human action recognition using kinect data," in *MultiMedia Modeling '14* eds C. Gurrin, F. Hopfgartner, W. Hurst, H. Johansen, H. Lee, and N. O'Connor (Dublin: Springer International Publishing), 473–483.

Parisi, G. I., and Wermter, S. (2013). "Hierarchical SOM-based detection of novel behavior for 3D human tracking," in *The 2013 International Joint Conference on Neural Networks* (Dallas, TX: IEEE), 1380–1387.

Parisi, G. I., Barros, P., and Wermter, S. (2014a). "FINGeR: framework for interactive neural-based gesture recognition," in *ESANN14* 443–447.

Parisi, G. I., Jirak, D., and Wermter, S. (2014b). "HandSOM - neural clustering of hand motion for gesture recognition in real time," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN '14)* (Edinburgh: IEEE), 981–986.

Parisi, G. I., Weber, C., and Wermter, S. (2014c). "Human action recognition with hierarchical growing neural gas learning," in *ICANN14* (Helderberg: Springer), 89–96.

Parisi, G. I., and Wermter, S. (2015). Neurocognitive assistive robot for robust fall detection. *Smart Environ.*

Pascual-Leone, A., Freitas, C., Oberman, L., Horvath, J. C., Halko, M., Eldaief, M., et al. (2011). Characterizing brain cortical plasticity and network dynamics across the age-span in health and disease with TMS-EEG and TMS-fMRI. *Brain Topogr.* 24, 302–315. doi: 10.1007/s10548-011-0196-8

Pavlova, M., and Sokolov, S. (2000). Orientation specificity in biological motion perception. *Percept. Psychophys.* 62, 889–899. doi: 10.3758/BF03212075

Pavlova, M., Krageloh-Mann, I., Sokolov, A., and Birbaumer, N. (2001). Recognition of point-light biological motion displays by young children. *Perception* 30, 925–933. doi: 10.1068/p3157

Perrett, D. I., Rolls, E. T., and Caan, W. (1982). Visual neurons responsive to faces in the monkey temporal cortex. *Exp. Brain Res.* 47, 329–342. doi: 10.1007/BF00239352

Rougier, C., Auvinet, E., Rousseau, J., Mignotte, M., and Meunier, J. (2011). "Fall detection from depth map video sequences," in *Toward Useful Services for Elderly and People with Disabilities*, eds B. Abdulrazak, S. Giroux, B. Bouchard, H. Pigot, and M. Mokhtari (Montreal, QC: Springer Berlin - Heidelberg), 121–128.

Shan, J., and Akella, S. (2014). "3D Human action segmentation and recognition using pose kinetic energy," in *Workshop on Advanced Robotics and its Social Impacts (ARSO14)* (Evanston, IL: IEEE), 69–75.

Shiffrar, M., and Freyd, J. J. (1990). Apparent motion of the human body. *Psychol. Sci.* 1, 257–264. doi: 10.1111/j.1467-9280.1990.tb00210.x

Singer, J. M., and Sheinberg, D. L. (2010). Temporal cortex neurons encode articulated actions as slow sequences of integrated poses. *J. Neurosci.* 30, 3133–3145. doi: 10.1523/JNEUROSCI.3211-09.2010

Sofatzis, R. J., Afshar, S., and Hamilton, T. J. (2014). "Rotationally invariant vision recognition with neuromorphic transformation and learning networks," in *2014 IEEE International Symposium on Circuits and Systems* (Melbourne, VIC: IEEE), 469–472.

Soltoggio, A., Lemme, A., Reinhart, F., and Steil, J. (2013a). Rare neural correlations implement robotic conditioning with delayed rewards and disturbances. *Front. Neurorobot.* 7:6. doi: 10.3389/fnbot.2013.00006

Soltoggio, A., Reinhart, F., Lemme, A., and Steil, J. (2013b). "Learning the rules of a game: neural conditioning in human-robot interaction with delayed rewards," in *IEEE International Conference of Development and Learning and on Epigenetic Robotics* (Osaka).

Stanley, J. C. (1976). Computer simulation of a model of habituation. *Nature* 261, 146–148. doi: 10.1038/261146a0

Strickert, M., and Hammer, B. (2005). Merge SOM for temporal data. *Neurocomputing* 64, 39–71. doi: 10.1016/j.neucom.2004.11.014

Suarez, J., and Murphy, R. (2012). "Hand gesture recognition with depth images: a review," in *RO-MAN14* (Edinburgh, UK: IEEE), 411–417.

Sumi, S. (1984). Upside-down presentation of the Johansson moving light-spot pattern. *Perception* 13, 283–302. doi: 10.1068/p130283

Sung, J., Ponce, C., Selman, B., and Saxena, A. (2012). "Unstructured human activity detection from RGBD images," in *2012 IEEE International Conference on Robotics and Automation* (Saint Paul, MN: IEEE), 842–849.

Tan, C., Singer, J. M., Serre, T., Sheinberg, D., and Poggio T. A. (2013). "Neural representation of action sequences: how far can a simple snippet-matching model take us?" in *Advances in Neural Information Processing Systems, NIPS13* (Lake Tahoe, CA: Harrahs and Harveys), 1–9.

Thirkettle, M., Benton, C. P., and Scott-Samuel, N. E. (2009). Contributions of form, motion and task to biological motion perception. *J. Vis.* 9, 1–11. doi: 10.1167/9.3.28

Thompson, R. F., and Spencer, W. A. (1966). Habituation: a model phenomenon for the study of neuronal substrates of behavior. *Psychol. Rev.* 73, 16–43. doi: 10.1037/h0022681

Thornton, I. M., Rensink, R. A., and Shiffrar, M. (2002). Active versus passive processing of biological motion. *Perception* 31, 837–853. doi: 10.1068/p3072

Troje, N. F. (2002). Decomposing biological motion: a framework for analysis and synthesis of human gait patterns. *J. Vis.* 2, 371–387. doi: 10.1167/2.5.2

Tyler, S. C., and Grossman, E. D. (2011). Feature-based attention promotes biological motion recognition. *J. Vis.* 11, 1–6. doi: 10.1167/11.10.11

Vangeneugden, J., Pollick, F., and Vogels, R. (2009). Functional differentiation of macaque visual temporal cortical neurons using a parametric action space. *Cereb. Cortex* 19, 593–611. doi: 10.1093/cercor/bhn109

von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik* 14, 85–100. doi: 10.1007/BF00288907

Weinland, D., Ronfard, R., and Boyer, R. (2011). A survey of vision-based methods for action representation, segmentation and recognition. *Comput. Vis. Image Underst.* 115, 224–241. doi: 10.1016/j.cviu.2010.10.002

Xu, R., Agarwal, P., Kumar, S., Krovi, V. N., and Corso, J. J. (2012). "Combining skeletal pose with local motion for human activity recognition," in *Articulated Motion and Deformable Objects (LNCS 7378)*, eds F. J. Perales, R. B. Fisher, and T. B. Moeslund (Mallorca: Springer Berlin Heidelberg) 114–123.

Yanik, P. M., Manganelli, J., Merino, J., Threatt, A. L., Brooks, J. O., Evan Green, K., et al. (2014) A gesture learning interface for simulated robot path shaping with a human teacher. *IEEE Trans. Hum. Mach. Syst.* 44, 44–51. doi: 10.1109/TSMC.2013.2291714

Zhang, C., and Y. Tian. (2012). RGB-D camera-based daily living activity recognition. *J. Comput. Vis. Image Process.* 2, 1–7.

Zhu, Y., Chen, W., and Guo, G. (2014). Evaluating spatiotemporal interest point features for depth-based action recognition. *Image Vis. Comput.* 32, 453–464. doi: 10.1016/j.imavis.2014.04.005

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: 22 July 2015
doi: 10.3389/fnbot.2015.00006

Learning touch preferences with a tactile robot using dopamine modulated STDP in a model of insular cortex

Ting-Shuo Chou[1]*, *Liam D. Bucci*[2] *and Jeffrey L. Krichmar*[1, 2]

[1] Department of Computer Sciences, University of California, Irvine, Irvine, CA, USA, [2] Department of Cognitive Sciences, University of California, Irvine, Irvine, CA, USA

Neurorobots enable researchers to study how behaviors are produced by neural mechanisms in an uncertain, noisy, real-world environment. To investigate how the somatosensory system processes noisy, real-world touch inputs, we introduce a neurorobot called CARL-SJR, which has a full-body tactile sensory area. The design of CARL-SJR is such that it encourages people to communicate with it through gentle touch. CARL-SJR provides feedback to users by displaying bright colors on its surface. In the present study, we show that CARL-SJR is capable of learning associations between conditioned stimuli (CS; a color pattern on its surface) and unconditioned stimuli (US; a preferred touch pattern) by applying a spiking neural network (SNN) with neurobiologically inspired plasticity. Specifically, we modeled the primary somatosensory cortex, prefrontal cortex, striatum, and the insular cortex, which is important for hedonic touch, to process noisy data generated directly from CARL-SJR's tactile sensory area. To facilitate learning, we applied dopamine-modulated Spike Timing Dependent Plasticity (STDP) to our simulated prefrontal cortex, striatum, and insular cortex. To cope with noisy, varying inputs, the SNN was tuned to produce traveling waves of activity that carried spatiotemporal information. Despite the noisy tactile sensors, spike trains, and variations in subject hand swipes, the learning was quite robust. Further, insular cortex activities in the incremental pathway of dopaminergic reward system allowed us to control CARL-SJR's preference for touch direction without heavily pre-processed inputs. The emerged behaviors we found in this model match animal's behaviors wherein they prefer touch in particular areas and directions. Thus, the results in this paper could serve as an explanation on the underlying neural mechanisms for developing tactile preferences and hedonic touch.

Keywords: tactile robot, reinforcement learning, dopamine, STDP, insular cortex, somatosensory cortex

OPEN ACCESS

Edited by:
Andrea Soltoggio,
Loughborough University, UK

Reviewed by:
Mehdi Khamassi,
Université Pierre et
Marie Curie, France
Felix Reinhart,
Bielefeld University, Germany

Correspondence:
Ting-Shuo Chou,
Department of Cognitive Sciences,
University of California, Irvine, 2224
Social & Behavioral Sciences
Gateway, Irvine, CA 92697-5100, USA
tingshuc@uci.edu

Received: *23 March 2015*
Accepted: *02 July 2015*
Published: *22 July 2015*

Citation:
Chou T-S, Bucci LD and Krichmar JL (2015) Learning touch preferences with a tactile robot using dopamine modulated STDP in a model of insular cortex. Front. Neurorobot. 9:6. doi: 10.3389/fnbot.2015.00006

Introduction

Humans and other animals respond preferentially to different types of touches. For example most cats prefer to be petted from head to tail rather than the other way around. Although, tactile sensing is an active area of robotics research, which takes inspiration from biology and neuroscience, most tactile robots have been developed to sense the borders and shapes of objects (Pearson et al., 2011; Evans et al., 2012; Schroeder and Hartmann, 2012) or for grasping and detecting surfaces (Bologna et al., 2011, 2013; Spigler et al., 2012). The present paper introduces a tactile neurorobot that has a surface designed for petting. The robot has the ability to signal its preferences through coloration of its surface and auditory signals. We will use this neurorobot to explore neural mechanisms of learning in uncertain, real-world environments.

In a set of mutual reinforcement learning experiments, we demonstrate that a user can pair colors on the robot's surface with hand sweeps in preferred directions across the robot's surface. The spatiotemporal nature of tactile stimuli, as well as the noisy sensors and environments, in which they operate, make the perception of touch a complex problem. To address these issues, we introduce a biologically spiking neural network, which learns through a novel dopaminergic spike timing dependent plasticity mechanism. Specifically, the robot has built-in tactile preferences and a user must learn these preferences, as well as reward the robot by touching the robot in its preferred ways. Auditory tones were used to signal pleasure and disappointment. In this way, the robot can learn the association between a color and a gentle touch.

Because the neurorobot's main sensory modality was touch, we developed a neurobiologically plausible model of tactile sensing. Mammals have two tactile pathways; one which is fast and delivers fine touch resolution, and another which is slower with coarser resolution that delivers hedonic or value-laden touch information. It is this latter touch pathway that we will explore in the present experiments. In animals, sophisticated cutaneous mechanoreceptors in the skin are capable of perceiving temperature, indentation, stretch, vibration, and movement (Abraira and Ginty, 2013). Unlike primary visual cortex for visual information, primary somatosensory cortex (S1) is not the only first order cortical region processing tactile information from thalamus (TH). Insular cortex (IC), which was thought to be higher hierarchical cortical region receiving tactile information from secondary somatosensory cortex (S2) (Felleman and Van Essen, 1991), also processes tactile information ascending directly from posterior ventromedial thalamus (VMpo) in macaque (Sewards and Sewards, 2002) and human (Craig et al., 1994). The parallel pathways to insular cortex in mammals imply that a single piece of tactile information could be heterogeneously processed in different regions and then integrated in insular cortex. For instance, a gentle touch detected by mechanoreceptors with C-fiber and Aβ-fiber triggers spike trains going through TH→IC and TH→S1→S2→IC pathways, respectively. The spike trains invoke pleasant sensation, which is correlated with insular activity (Morrison et al., 2011a,b). The pleasant sensation could be a state of emotional representation in anterior insular cortex (AIC) as a result of integrating tactile information along posterior insular (pIC), mid-insular (mIC) and anterior insular (AIC) (Craig, 2002, 2009). In the present study, we are interested in the neural mechanism for integrating tactile information in this area because the unconscious element of the pleasant sensation might link to the dopamine system (Schultz, 2006) and therefore defines innate preferences or values (Krichmar and Rohrbein, 2013).

To explore learning mechanisms for hedonic touch, we constructed a spiking neural network (SNN) model of the posterior insular cortex (pIC), somatosensory cortex, and the areas necessary for value-based learning. The neural dynamics in the model of pIC accounted for: (1) the robot's tactile preferences, (2) processing real-world tactile inputs with minimal pre-processing, (3) demonstrating that wave propagation is a viable means to generating precise spike timing in the face of noise, and (4) learning associations between neutral stimuli and hedonic touch.

Because hedonic touch requires a caresser and a caressee, we developed a human robot interaction study that required mutual reinforcement learning. To achieve these goals, we built a robot, named CARL-SJR (Cognitive Anteater Robotics Laboratory—Spike Judgment Robot), with a large tactile sensory area and a surface capable of displaying bright colors. CARL-SJR's behavior was controlled by the dual-pathway model (Brown et al., 1999; Tan and Bullock, 2008; Chorley and Seth, 2011), which minimizes prediction error signaled by dopamine (Schultz, 2006).

Materials and Methods

CARL-SJR Neurorobot

The sensory encoding and learning experiments were conducted with a novel robot named CARL-SJR. CARL-SJR is autonomous, mobile, self-contained, and capable of tactile sensing and interaction (see **Figure 1**). To give the robot a sense of touch, we incorporated an array of trackballs, which are typically found in cellphones and other devices. The trackball array can signal the direction and velocity of tactile stimuli. The robot's unique form factor encourages users to rub or pet its surface. The robot has LEDs co-located at each trackball, which can display a wide range of colors in response to touch. CARL-SJR has a large tactile sensory area and the ability to display bright colors on its shell (see **Figure 1A**). CARL-SJR's shell has a 9-by-8 matrix of true color LEDs. Any animated color pattern can be programmed to display on its shell. The shell also has a 9-by-8 matrix of trackballs, which are used for sensing tactile input. The trackballs form a coordinate system with the upper left trackball mapped to the origin (0, 0), and the downward right trackball mapped to the maximum coordinates (8, 6) (see **Figure 1B**). A trackball can detect a touch event in four directions (i.e., up, down, left, and right). Assuming a trackball rolls in the left direction, a sequence of touch events will be generated as shown by the timeline in **Figure 1C**. In this example, the majority of touch events are in the left direction accompanied by noise in other directions. The current version of CARL-SJR mounted the tactile shell on an iRobot Create platform. A computer, which communicated with CARL-SJR over Bluetooth, executed the neural model, collected trackball data, controlled the LEDs on the shell, and controlled the motors and speakers on the iRobot Create. More details on the robot hardware can be found at Bucci et al. (2014).

CARL-SJR displayed color patterns as output, and took hand movements as input. The display patterns could be a solid color, mixed, or animated. **Figure 1D** shows an example of an animated

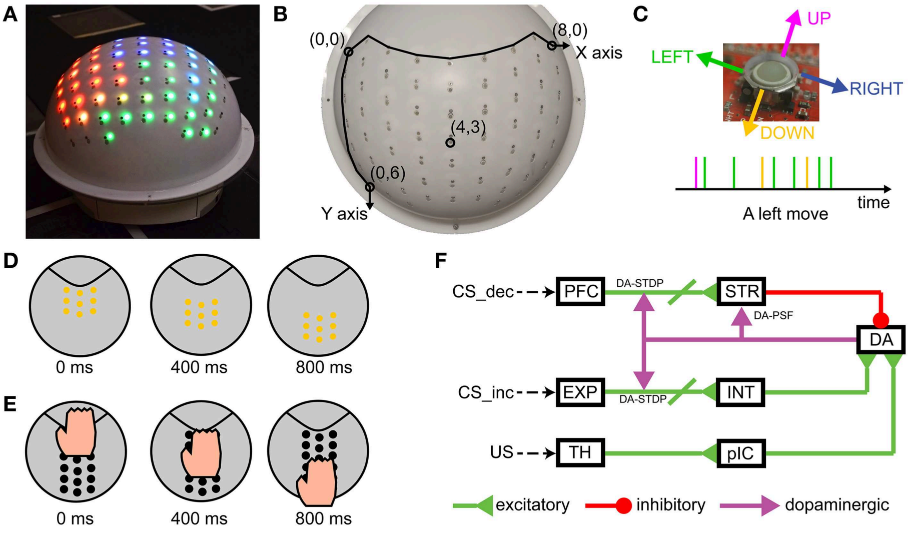


FIGURE 1 | CARL-SJR is an interactive, tactile neurorobot. (A) Photograph of CARL-SJR. The shell has a 9-by-8 matrix of trackballs each collocated with red, green, and blue color LEDs. Any animated color pattern is possible. **(B)** The 9-by-8 matrix of trackballs forms the coordinate system of CARL-SJR. The most up left trackball is mapped to the origin (0, 0) while the most down right trackball is mapped to the coordinates (8, 6). **(C)** A trackball can detect a touch event in four directions. Assuming a trackball rolls in the left direction, a sequence of touch events will be generated as shown by the timeline. The majority of touch events will be left events with events in other directions due to sensor noise. **(D)** An example of an animated color pattern. The yellow pattern moves downward in 800 ms. **(E)** An example of a typical touch pattern. The hand moves downward. Usually 4 ∼ 7 trackballs are touched simultaneously. **(F)** Schematic of the spiking neural network architecture that controlled CARL-SJR's behavior.

color pattern. The yellow pattern moves downward in 800 ms. Hand movements across the shell triggered touch events in the matrix of trackballs. **Figure 1E** shows a typical touch pattern. The hand moves downward. Usually 4 ∼ 7 trackballs are touched simultaneously.

Spiking Neural Network Model

To support the present mutual reinforcement learning experiments, we built a spiking neural network (SNN) model using the large scale SNN simulator CARLsim to recognize tactile sensory input, and to control CARL-SJR's behavior (Nageswaran et al., 2009; Richert et al., 2011; Carlson et al., 2014). CARLsim was written in C/C++/CUDA and designed to leverage the parallel computing power of GPUs. The present SNN model had 13,000 neurons and 200,000 synapses and could run four times faster than real time. However, the model was slowed down to match the real time robotic application.

The SNN was designed to be biologically plausible and simulated somatosensory pathways, as well as neurally inspired learning. To support learning, we implemented a variation of the dual-pathway model (Brown et al., 1999; Tan and Bullock, 2008; Chorley and Seth, 2011), which minimizes prediction error signaled by dopamine (Schultz, 2006). In the present experiments, the Conditioned Stimulus (CS) was a color pattern displayed on CARL-SJR's surface, and the Unconditioned Stimulus (US) was a touch pattern initiated by the user sweeping his or her hand across CARL-SJR's surface. The decremental (dopamine) pathway for the CS (see PFC→STR→DA in **Figure 1F**) decreased DA neurons' activity through inhibitory projections. In contrast, the incremental (dopamine) pathway for the US (see TH→pIC→DA in **Figure 1F**) increased spikes of dopaminergic neurons. The complementary pathways converge on a group of DA neurons and control the DA response. Phasic neural activity in the incremental pathway for US might change the balance of excitation and inhibition, thus triggering a DA burst, which in turn signals striatum (STR) and PFC→STR synapses through dopaminergic projections. The decremental pathway is able to learn the timing of US and then increases inhibitory force on DA neurons for restoring the balance. The neural activities in prefrontal cortex (PFC) and striatum are crucial for learning the timing of US. Chorley and Seth's model incorporated pre-generated polychronous groups (Izhikevich, 2006) for precisely timed spikes in PFC. However, polychronous groups are a theoretical prediction and, to the best of our knowledge, have not been shown empirically. Moreover, it would be difficult to show repeatability of this precise timing in a computational model having noisy and uncertain inputs. Rather than relying on precisely timed polychrony or synfire chains, we used wavelike neural activity for propagating information through the simulated brain regions. These waves of neural activity have empirical support and do not require precisely timed spike sequences (Rubino et al., 2006; Benucci et al., 2007; Ferezou et al., 2007; Han et al., 2008; Wu et al., 2008; Lubenov and Siapas, 2009; Sato et al., 2012).

Spiking Neuron Model

CARLsim incorporated a phenomenological model of a spiking neuron proposed by Izhikevich (2003). The dynamics of each neuron is governed by the following equations:

$$\dot{v} = 0.04v^2 + 5v + 140 - u + I \tag{1}$$

$$\dot{u} = a\left(bv - u\right) \tag{2}$$

The variable v is the membrane potential of a neuron and the variable u is an abstract membrane recovery current. The variable I is the input current (i.e., the current flow into a neuron). A neuron emits a spike if its membrane potential is higher than 30 mv and then resets according to the following equation:

$$if\ v \geq 30,\ then \begin{cases} v = c \\ u = u + d \end{cases} \tag{3}$$

Both excitatory regular spiking (RS) neurons and inhibitory fast spiking (FS) neurons were used in the model. For RS neurons, we set $a = 0.02$, $b = 0.2$, $c = -65.0$, and $d = 8.0$. For FS neurons, we set $a = 0.1$, $b = 0.2$, $c = -65.0$, and $d = 2.0$. For more biologically realistic dynamics, a conductance synapse model was used to calculate the input current for each neuron (Izhikevich and Edelman, 2008). The equation is:

$$\begin{aligned} I &= g_{AMPA}\left(0 - v\right) + g_{NMDA}\frac{\left[\frac{-80-v}{60}\right]^2}{1 + \left[\frac{-80-v}{60}\right]^2}\left(0 - v\right) \\ &+ g_{GABA_A}\left(-70 - v\right) + g_{GABA_B}(-90 - v) \end{aligned} \tag{4}$$

where v is again the membrane potential and g is the total conductance for ion channels created by different receptors (i.e., AMPA, NMDA, GABA_A, GABA_B). The conductance g is increased by the amount of synaptic weight w upon the arrival of a spike and decays along time as described by the equations below:

$$g_{i,\,k} = \sum_{j}^{N} w_{jk}\delta(t - t_{pre,\,j}),\ i \in \{AMPA,\ NMDA,\ GABA_A,\ GABA_B\} \tag{5}$$

$$\dot{g}_{i,\,k} = -\frac{g_{i,\,k}}{\tau_i}\ i \in \{AMPA,\ NMDA,\ GABA_A,\ GABA_B\} \tag{6}$$

where N is the number of pre-synaptic neurons and w_{jk} is the weight of the synapse connecting pre-synaptic neuron j and post-synaptic neuron k. δ is the Dirac delta function. t is the current time (i.e., current simulation time step when we approximate the continuous function in discrete time steps) and $t_{pre,\,j}$ is the arrival time of the last spike from neuron j. The decay constant τ_i was set to 5, 100, 6, and 150 ms for different receptors AMPA, NMDA, GABA_A, and GABA_B, respectively.

STDP, DA-STDP and DA-PSF

Spike timing dependent plasticity (STDP) (Caporale and Dan, 2008; Markram et al., 2011) was applied to excitatory and inhibitory synapses in the computational model. In our model, excitatory STDP and inhibitory STDP were used to develop and stabilize wavelike neural activity in the PFC area (see Section Wave Propagation in PFC). The synaptic weights were governed by the following equations:

$$\dot{w}_{exc} = A^+ e^{\frac{t_{pre} - t_{post}}{\tau_+}}\delta(t - t_{post}) - A^- e^{\frac{t_{post} - t_{pre}}{\tau_-}}\delta(t - t_{pre}) \tag{7}$$

$$\begin{aligned} \dot{w}_{inh} &= B^+H^+\left(\left|t_{post} - t_{pre}\right|\right)\delta\left(t - t_{pre}\right) \\ &+ B^+H^+\left(\left|t_{post} - t_{pre}\right|\right)\delta\left(t - t_{post}\right) \\ &- B^-H^-\left(\left|t_{post} - t_{pre}\right|\right)\delta\left(t - t_{pre}\right) \\ &- B^-H^-\left(\left|t_{post} - t_{pre}\right|\right)\delta\left(t - t_{post}\right) \end{aligned} \tag{8}$$

$$H^+\left(x\right) = \begin{cases} 1,\ if\ 0 < x \leq \lambda \\ 0,\ otherwise \end{cases},\ H^-\left(x\right) = \begin{cases} 1,\ if\ \lambda < x \leq \gamma \\ 0,\ otherwise \end{cases} \tag{9}$$

The variable w_{exc} is an excitatory synaptic weight. t_{pre} is the arrival time of last pre-synaptic spikes while t_{post} is the time of post-synaptic spikes. δ is again the Dirac delta function. We set the E-STDP parameters A^+/A^- and τ_+/τ_- to 0.1/0.07 and 20/40 ms respectively. The variable w_{inh} is an inhibitory synaptic weight. The I-STDP were modeled as a piecewise linear Mexican hat function as was described in Srinivasa and Jiang (2013). The value of $|t_{pre}$-$t_{post}|$determines LTP or LTD. λ and γ define the ranges of inhibitory LTP and LTD. We set the I-STDP parameters B^+/B^- and λ/γ to 0.1/0.06 and 4/20 ms.

Dopamine modulated spike timing dependent plasticity (DA-STDP) served as the underlying neural mechanism for reinforcement learning and solving the distal reward problem (Izhikevich, 2007). Both the incremental and decremental pathways for CS receive dopamine signals (see **Figure 1F**) and their synapses are subject to DA-STDP. In this form, the E-STDP function does not directly change synaptic weights, but instead modulates weights through an eligibility trace. The change of eligibility trace c, dopamine value d, and excitatory synaptic weight w_{exc} are described by the following equations:

$$\begin{aligned} \dot{c} &= -c/\tau_c + A^+ e^{\frac{t_{pre} - t_{post}}{\tau_+}}\delta(t - t_{post}) \\ &- A^- e^{\frac{t_{post} - t_{pre}}{\tau_-}}\delta(t - t_{pre}) \end{aligned} \tag{10}$$

$$\dot{d} = -d/\tau_d + da_{syn}\delta(t - t_{pre}) \tag{11}$$

$$\dot{w}_{exc} = cd \tag{12}$$

The excitatory synaptic weight w_{exc} is scaled by variable d, which is the dopamine concentration of the target neural group (i.e., the post neural group projected by dopaminergic synapses). The dopamine value d is increased by da_{syn}, which is 0.04, for each spike reaching the target neural group. The value of d ranges from the baseline value 1.0 μM to a peak value 20.0 μM. Please note, both d and c decay over time. The symbol δ represents the Dirac delta function and is one if there is an action potential at t_{pre} and zero otherwise. The time constant τ_c and τ_d are 1000 and 50 ms as shown in Equations (10) and (11), respectively.

Dopamine modulated post-synaptic facilitation (DA-PSF) is the phenomenon where excitability of a post-synaptic neural group is modulated by dopamine (Nicola et al., 2000; Williams and Castner, 2006). A large portion of medium spiny neurons in striatum has D1 receptors to allow extra input current. We modeled this phenomenon as the following equation:

$$g_{efc,\,i} = g_i\left(0.9 + 0.1d\right),\ i \in \{AMPA,\ NMDA\} \tag{13}$$

where d is the dopamine value and g_{efc} is the effective conductance used for calculating input current by Equation (4). Note that the minimum gain 1.0 is due to the baseline dopamine value of 1.0 μM.

Network Architecture

The neural architecture for the present study, which is based on the dual-pathway model (Brown et al., 1999; Tan and Bullock, 2008; Chorley and Seth, 2011), is depicted in **Figure 2A**. Color information, from CARL-SJR's shell, which represents the CS, is fed into both the decremental and incremental pathways. The incremental pathway also signals tactile information, which represents the US. It has been suggested that the incremental and decremental pathways balance each other out so that after learning the dopamine signal is suppressed when a reward is predicted by the CS and arrives at the expected time with a US.

The decremental pathway for the CS goes from the ActionGenerator (AG), to the prefrontal cortex (PFC), to the striatum (STR), and then to a pool of dopaminergic neurons (DA). All connections in the decremental pathway are excitatory, except for the STR→DA connections, which are inhibitory. The cortical area AG is used to generate spike trains that encode color responses. AG has 64 excitatory neurons for four CS input channels (i.e., red, green, blue, and yellow). Each neuron in a channel, which has 16 neurons totally, emits a spike if the corresponding CS input is presented. **Figure 2B** illustrates more detailed connections of a channel from AG to PFC to STR. A pre-synaptic neuron in AG is topographically connected to a post-synaptic neuron in PFC based on the distance between neuron locations (see x_dist and y_dist in **Figure 2B**). The smaller green circle in **Figure 2C** defines the Excitatory Forward Projecting Radius (EFPR), which is the standard deviation of the Gaussian projection in spatial domain. Specifically, the distance between two neurons determines the connection probability (i.e., Gaussian distribution) and conductance delay (i.e., axonal delay). A channel in PFC has 2048 excitatory neurons and 512 inhibitory neurons. They are connected laterally according to Excitatory/Inhibitory Lateral Projecting Radius (ELPR/ILPR). The excitatory/inhibitory radius and synaptic weights were tuned to exhibit the behavior of traveling waves (Chen et al., 2013). If a CS input is presented, AG transmits spikes to the left side of the corresponding channel in PFC. The wide rectangular shape allows wavelike neural activity to propagate along the long side with precise timing. The idea behind the wave propagation in PFC is that the position of a neuron encodes the time relative to the release of a CS inputs (see Section Wave Propagation in PFC). PFC and STR are topologically connected according to EFPR as well. PFC→STR synaptic weights were subject to DA-STDP and

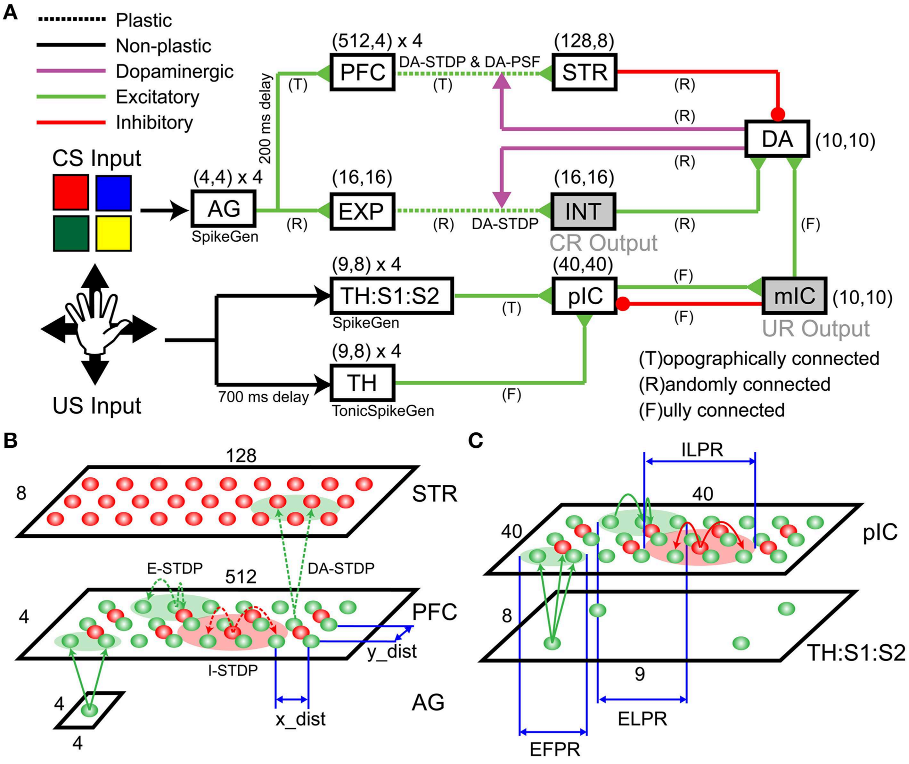


FIGURE 2 | Detailed Spiking Neural Network (SNN) architecture. The SNN network model includes incremental (dopamine) pathway and decremental (dopamine) pathway. **(A)** An overview of network architecture. The model consists of an Action Generator (AG) group, a sensory experience (EXP) group, prefrontal cortex (PFC), striatum (STR), an intermediate area (INT), thalamus (TH), thalamus/somatosensory area (TH:S1:S2), posterior insular cortex (pIC), mid-insular cortex (mIC), and a group of dopaminergic neurons (DA). **(B)** Schematic of the projection patterns in the incremental pathway from AG to PFC to STR. **(C)** Schematic of the projection patterns in the decremental pathway from TH:S1:S2 to pIC.

sensitive to DA signals. The plastic PFC→STR synapses were able to learn (or predict) the timing of following US (see Section PFC-to-STR Synapses Learn to Predict the Timing of US). STR and DA are randomly connected and STR→DA synaptic weights were tuned to match the range of excitatory force from the incremental pathways.

The incremental pathway for the CS goes from AG to cortical area Expression (EXP), to Intermediate area (INT), and then to DA with excitatory projections. AG→EXP→INT→DA are randomly connected. EXP→INT connections are plastic synapses subject to DA-STDP and representations for CS inputs (i.e., red, green, blue, and yellow) will form in INT if a color pattern is reinforced (see Section CARL-SJR's Behaviors during Learning Multiple CS-US Pairs). INT→DA synaptic weights were tuned to match the range of inhibitory force from STR. INT was also linked to conditioned response (CR) in which CARL-SJR rotates its body if INT has more than 50 spikes within 50 ms.

For the incremental pathway, we implemented both the fast Aβ and the slower C-fiber tactile pathways. The incremental pathway for US starts with tactile inputs from touch events and projects from the parallel thalamus/thalamus:somatosensory cortex (TH/TH:S1:S2) paths, to the posterior insular cortex (pIC), to the medial insular cortex (mIC) and then to DA. The TH and TH:S1:S2 have four US input channels (i.e., up, down, left, and right). A touch event in **Figure 1C** is one-to-one mapped to a spike in the corresponding channel in TH:S1:S2 where 9-by-8 neurons for each channel match the layout of trackballs. The spikes going through the TH:S1:S2→pIC pathway represent tactile information carried by the fast Aβ-fiber. To model fast spike transmission and acuteness in spatial resolution of this pathway, we topographically connected TH:S1:S2 and pIC as shown in **Figure 2C** without any delay. In contrast, the spikes going through TH→pIC pathway represent tactile information carried by the slower C-fiber pathway, where a touch event is mapped to a period of tonic spikes. We delayed spike generation in TH by 700 ms to simulate the slower conduction speed of the C-fiber pathway, and fully connected TH to pIC to simulate poor spatial resolution. Since we hypothesize that a piece of tactile information is heterogeneously processed through the parallel paths and then integrated in pIC, the synaptic weights of TH:S1:S2→pIC and TH→pIC were tuned to fulfill the condition that neither TH:S1:S2→pIC nor TH→pIC can dominate the neural activity in pIC. Specifically, the neural activity in pIC was strong enough to drive neural response in mIC and then DA only when there was neural activity in both TH:S1:S2 and TH. As a result, the excitatory force on DA neurons reflects the integration of the neural activity in pIC. To suppress neural activity in pIC after mIC is signaled by pIC, we added feedback connections from mIC to inhibitory neurons in pIC to simulate the effect of shunting inhibition (Silver, 2010). The mIC is linked to the unconditioned response (UR), in which CARL-SJR sings a high tone if mIC has more than 30 spikes within 50 ms.

These complementary excitatory and inhibitory pathways converge on DA neurons. The interaction among STR, INT, and pIC activities controls DA response. The DA responded with a burst (see Section Control CARL-SJR's Tactile Preference through Insular Cortex Model) when the excitatory force was larger, a DA dip (see Section Extinguishing Behaviors after Learning) when the inhibitory force was larger, or spontaneous activity when excitatory and inhibitory forces were balanced. DA is connected to STR and INT through dopaminergic projections. The dopamine values of STR and INT modulate PFC→STR and EXP→INT synapses by DA-STDP and DA-PSF as indicated in **Figure 2A**.

The synaptic weights, including initial, maximum, and minimum value, were tuned to match the number of pre-synaptic neurons and to prevent run-away neural dynamics. The conductance delays were tuned to maintain stable timing behaviors. The spontaneous firing rates in the cortical regions, ranging from 0.1 to 0.5 Hz (Griffith and Horn, 1966; Koch and Fuster, 1989), were tuned to support baseline neural activities, which are essential for STDP or DA-STDP. The complete network parameters are described in the Supplementary Materials.

Tactile Inputs

We analyzed the properties of tactile inputs (see **Figure 3**) to gain insight into tactile processing. **Figure 3A** shows raster plots and heat maps of three tactile inputs: (1) a downward hand sweep on CARL-SJR's left side, (2) a downward hand sweep in the middle, and (3) a rightward hand sweep in the middle. Since a touch event (see **Figure 1C**) was a one-to-one mapping to spikes in TH:S1:S2 area, the raster plots shows touch events as well. We can see the timing of spikes is irregular and noisy. As for heat maps, they show which trackballs were touched, as well as a somatotopic map in TH:S1:S2 area. Because of the curvature of CARL-SJR's surface, the trackballs were not aligned along the x-axis but mostly aligned along the y-axis (see **Figure 1B**). Thus, it was harder for users to touch complete rows of trackballs rather than columns of trackball.

Figure 3B shows distributions for touch duration, touch speed, number of touch events, and noise level over 1400 touch movements. The data of upward and downward movements were grouped in the first row while leftward and rightward movements were grouped in the second row because CARL-SJR's asymmetric trackball distribution affected the user's tactile inputs. The (mean, std) of touch durations were (935, 142) and (1,071, 128) ms for vertical and horizontal movement, respectively. The (mean, std) speeds were (7.3, 1.08) and (6.73, 0.93) inch/second for vertical and horizontal movement, respectively. The (mean, std) number of events were (207, 44) and (305, 69) for vertical and horizontal movement, respectively. The (mean, std) noise level were (8.31, 3.97%) and (19.72, 5.53%) for vertical and horizontal movement, respectively, where noise level was defined by the total number of direction events in an unexpected direction divided by the number of direction events in an expected direction. The noise level of horizontal movements is significantly higher due to CARL-SJR's curvature along y-axis.

Experimental Paradigm

The experiment paradigm is shown in **Figure 4**. A trial lasted 6 s. CARL-SJR initiated a trial by displaying a color on its surface

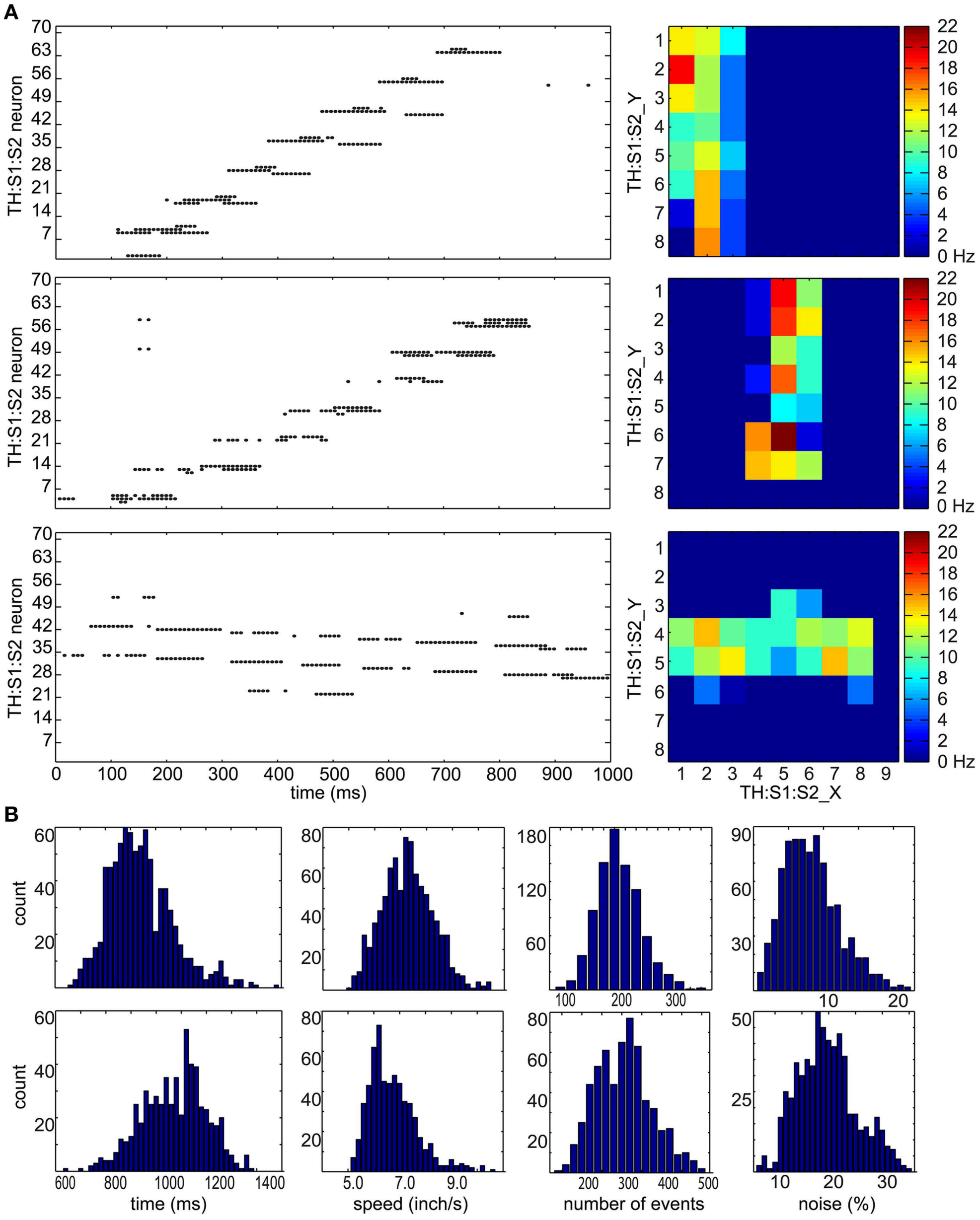


FIGURE 3 | Tactile stimulus analysis. (A) Representative spike raster plots and heatmaps of three tactile inputs to the TH:S1:S2 area. On the left are raster plots where the x-axis represents times in milliseconds, and the y-axis represents the neuron number. The top row shows an upward hand sweep on the left side of CARL-SJR's shell, the middle row shows an upward hand sweep in the middle of CARL-SJR's shell, and bottom row shows a rightward hand sweep in the middle of CARL-SJR's shell. The raster plots illustrate how sensory input leads to irregular and noisy spike activity in TH:S1:S2. On the right are heatmaps that show the mean neural activity corresponding to the raster plots. The x and y position in the heat map reflects the somatotopic organization of the TH:S1:S2 neurons. **(B)** The distribution of tactile inputs in contact duration, moving speed, number of events, and noise level (i.e., the amount of unexpected directional moves divided by the expected directional moves). The first row shows analysis based on 800 upward and downward moves while the second row shows analysis based on 600 leftward and rightward moves. The distributions of vertical and horizontal tactile input are quite different because locations of trackballs are asymmetric in vertical and horizontal direction.

(see Sections CARL-SJR's Behaviors during Learning Multiple CS-US Pairs and Extinguishing Behaviors after Learning for experimental details). A CS signal corresponding to the displayed color was input to the AG at 1.1 s after trial beginning. Effectively, neural activities were triggered in EXP (i.e., incremental pathway for CS) and PFC (i.e., decremental pathway) at 1.1 and 1.3 s respectively. It was the user's choice to reward CARL-SJR or not. If the user liked the displayed color, he/she could touch CARL-SJR within 2 s US window and the US signal (i.e., tactile input) was delivered to TH and TH:S1:S2, which in turn triggered

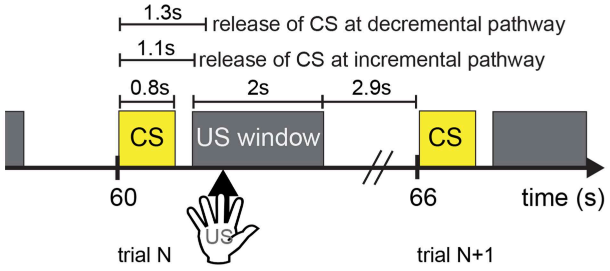

1.3s
release of CS at decremental pathway
1.1s
release of CS at incremental pathway
0.8s
2s
2.9s
CS
US window
CS
60
66
time (s)
US
trial N
trial N+1

FIGURE 4 | The timing diagram illustrates the experimental setup. Each trial lasts 6 s, where a CS signal is followed by a 2-s US window. It is a user's choice to provide a US signal (i.e., tactile input) or not to CARL-SJR. Tactile input outside the US window is unrewarded. After the US window has passed, there is a 2.9-s period before the next trial.

a dopaminergic reward response (see **Figure 2A**). Tactile input outside the US window did not activate this reward pathway. After the US window, there was a 2.9 s stabilizing period before the next trial. The experiment continued until learning achieved a desired level (e.g., the probability of CR is higher than UR).

Results

Control CARL-SJR's Tactile Preference through Insular Cortex Model

To characterize the network dynamics in response to sensory input, we ran a set of simulation experiments to explore the SNN's ability to transfer tactile information across different simulated brain regions. The incremental pathway for US was implemented by the model of pIC, which exhibited complicated neural dynamics in response to hand movements. We set TH→pIC synaptic weights to 0.04 for each channel and ran simulations with 400 upward movements as inputs to TH:S1:S2 and TH. **Figure 5A** illustrates representative neural activities in TH:S1:S2, TH, pIC, and DA. The raster plots in the first row show the spikes directly triggered by touch events in the upward direction. Green boxes indicate the tonic spikes generated by TH in the upward direction, which arrive at pIC 700 ms later because the conductance speed of C-fibers is slower than Aβ-fibers. Before the arrival of tonic spikes, pIC is at the state with low excitability. The spikes from TH:S1:S2 trigger local wavelike neural activity in pIC (see spikes earlier than green dashed lines in the second row). After the arrival of tonic spikes from TH, pIC transitions to a higher excitability state that can trigger global wavelike neural activities in the pIC (see spikes later than green dashed lines). Multiple waves may interact with others and may trigger a DA burst depending on the strength of the instant excitatory force (see histograms in the third row and first three columns for comparison). Note that, TH→pIC synaptic weights are crucial to the excitability of pIC, sustainability of global waves in pIC, and the probability of DA bursts.

The response of pIC was complex and influenced dopaminergic activity. Based on the DA response, we classified neural activities in pIC into five groups: (A1) strong DA burst, (A2) weak DA burst, (A3) no DA burst due to insufficient instant activity in pIC, (A4) no DA burst due to no global activity in pIC. (A5) multiple DA bursts. A gentle hand movement (with moderate speed) is most likely to trigger a DA burst. If the speed of a hand movement is too fast (e.g., the touch duration is less than 700 ms), the neural activity in TH:S1:S2 disappears before tonic spikes from TH tune pIC to excitable state and therefore pIC is unlikely to trigger a DA burst. On the other hand, if the speed is too slow, the excitatory force of TH:S1:S2 is too weak to trigger global waves in pIC even though pIC is tuned to excitable state by tonic spikes from TH.

We adjusted TH→pIC synaptic weights in different channels (i.e., up, down, left, and right) to control the probability of a DA burst linked to CARL-SJR's tactile preferences. To evaluate the probability of a DA burst against different TH→pIC synaptic weights, we ran simulations with 400 upward, 400 downward, 300 leftward, and 300 rightward movements as inputs to TH and TH:S1:S2. For each direction of inputs, we recorded the total number of single DA bursts and derived the burst probability (see **Figure 5B**). To make CARL-SJR prefer rightward movements, for example, we set the TH→pIC synaptic weights in right channel to 0.04, and the other channels to 0.03. In this case, the probability of a DA burst was greater than 70% for rightward movements but less than 20% for other directions. As a result, CARL-SJR learned faster if the user gives rewards by rightward movements. **Figure 5B** also shows how the asymmetric trackball distribution affects the number of touch events per hand movement. Note that there are typically more touch events for left and right movements than up and down movements (see the histograms at third column in **Figure 3B**). This can have an effect on CARL-SJR's tactile preferences. For example, if we set TH→pIC synaptic weights to 0.04 for each channel, horizontal movements yield slightly higher probability of generating a DA burst than vertical movements (see the black dash line in **Figure 5B**). Under the condition that TH→pIC synaptic weights for each channel are identical, CARL-SJR will prefer horizontal movements. In the experiments described below, the TH→pIC synaptic weights were set to 0.04 for the preferred direction, and 0.03 for the non-preferred direction. Taken together, these simulations showed that the simulated insular cortex could control dopamine bursts, which could lead to the shaping of tactile preferences.

Wave Propagation in PFC

A critical requirement for learning is assigning the credit of a reward to the appropriate stimulus that occurred in the past. Wave propagation has been suggested to be important for computing this timing in classical conditioning tasks (Palmer and Gong, 2014). The idea is that wavelike neural activity in the cortex might encode timing information related to events.

To investigate if wave propagation was a viable mechanism for the spatiotemporal learning in the present experiments, we incorporated this idea into our simulated PFC and demonstrated traveling waves in a series of simulations (see **Figure 6**). We modeled PFC as a long rectangular shape and tuned the conductance delay of lateral projections for excitatory neurons to be 15–20 ms and inhibitory neurons to be 1 ms. During a development period, we enabled E/I-STDP and delivered spikes

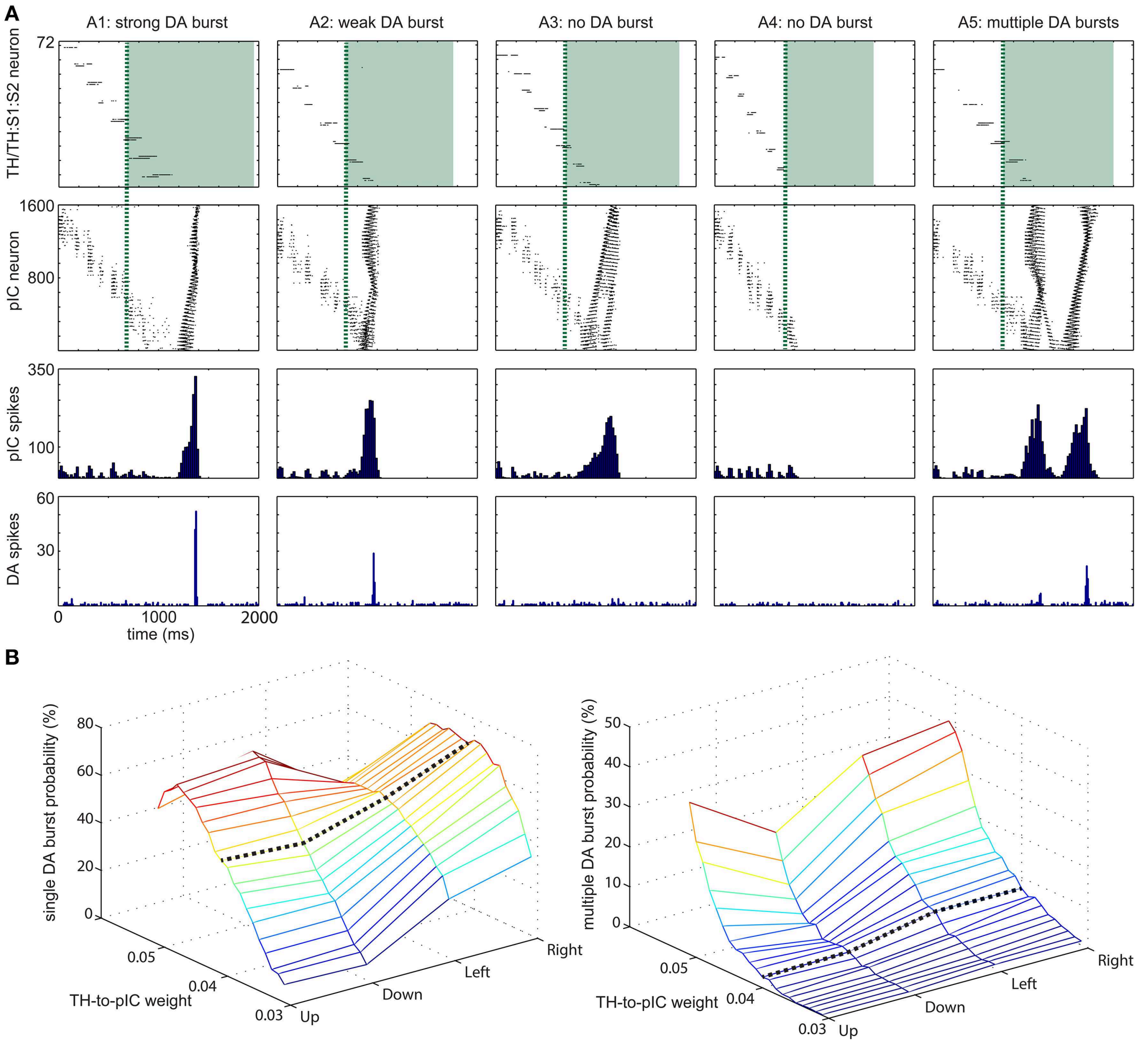


FIGURE 5 | Range of dopamine (DA) responses to varying input patterns (A) Five representative examples for DA response to tactile inputs. (A1) strong DA burst. (A2) weak DA burst. (A3) no DA burst due to insufficient instantaneous activity at pIC area. (A4) no DA burst due to no integration of activity at TH and TH:S1:S2 area. (A5) multiple DA bursts. The first row shows raster plots of TH:S1:S2 and TH areas. The spikes of TH area are drawn as green shade regions. The second and third rows show the magnitude of activity at pIC area, which leads to different DA response. The fourth row shows DA response. **(B)** TH→pIC synaptic weights affect DA response. (Left) The probability of a single DA burst based on synaptic weights and the moving direction of the tactile input. (Right) The probability of multiple DA bursts. Setting TH→pIC synaptic weight to 0.04 yields more than 50% of single DA burst while multiple DA burst is lower than 10%.

to the most left side of PFC every 4 s. After 2000 s of development, the wave reliably propagated from the left side to the right side of PFC in 2 s as shown by the raster plot in **Figure 6A.** The heat maps further show the location of a wave at a given time period. After starting the spike activity on the leftmost side of PFC, the wave reached the rightmost side of PFC at 1900 ms. The speed of waves is quite stable because of the lateral conductance delay. As a result, a neuron's location along x-axis encodes time information with high accuracy and reliability.

Noise might stop wave propagation as well as trigger unexpected waves. To address this issue, we tested the robustness of wave propagation under a noisy environment (see **Figure 6B**). Since we knew when a wave should arrive the most right side of PFC, we defined a fail case to be the absence of neural activity of the most right neurons [i.e., (508, y) ∼ (511, y)] at the expected time window (i.e., 1950 ∼ 2050 ms). We also defined a ghost wave case to be any occurrence of neural activity outside the expected time window. After the development period, the PFC was tested in 1000 trials where we delivered spikes to the most left neurons, which were connected to AG (see **Figure 2B**), and then recorded the number of fail cases and ghost wave cases. We derived the fail rate (in percentage) and ghost wave rate against different magnitudes of noise (in Hz) and the height of PFC (in the number of neurons). The fail rate was under 10% if the height of PFC was 4 neurons, and was under 40% if the height of PFC was 8 neurons. However, the fail rate was around 80% when the height

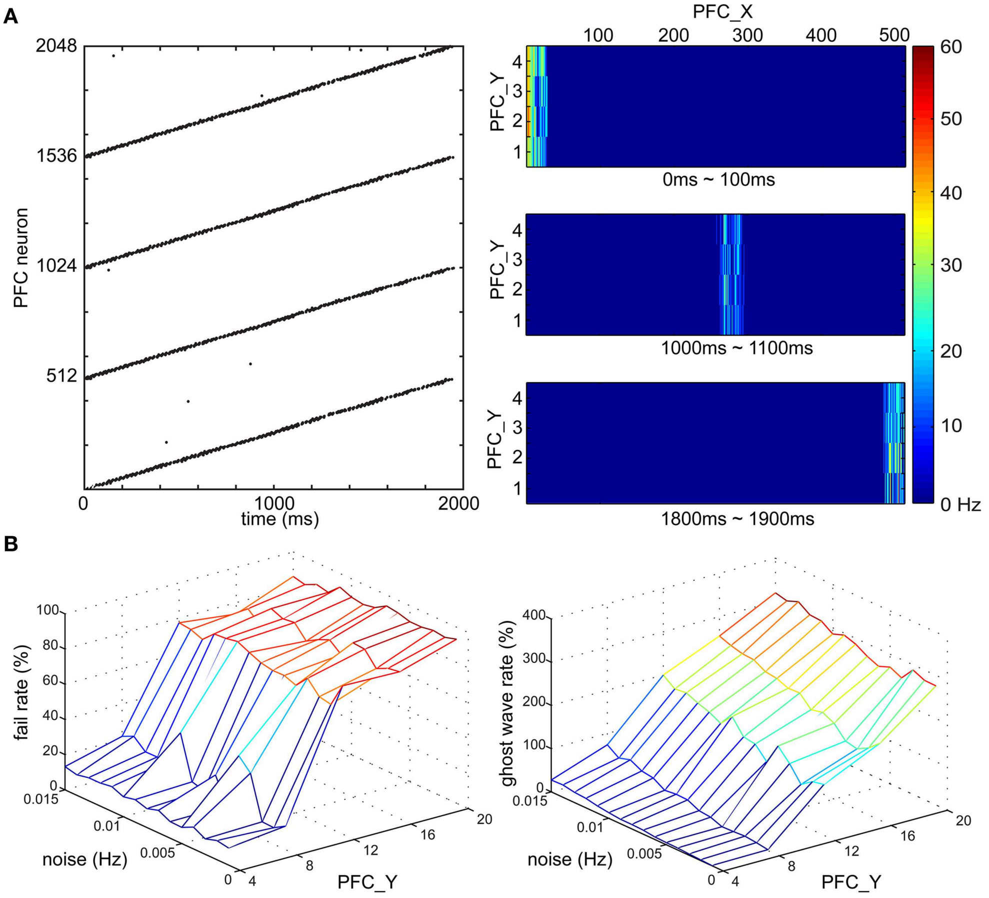


FIGURE 6 | Time course of PFC wave propagating activity. (A) Wave propagation in PFC area. (Left) The raster plot shows the time duration of the wave is around 2000 ms. (Right) the wave travels along x-axis of the PFC area and the locations of the wave at three time intervals, 0–100, 1000–1100, and 1800–1900 ms are shown from top to bottom, respectively. **(B)** The fail rate of a wave and the ghost wave rate based on noise (Hz) and the height of PFC area (i.e., the number of PFC neurons along y-axis). The noise level affects the probability of a wave reaching from one end of PFC to another, and on generating a ghost wave. PFC area is more resistant to noise if the height is smaller than 12 neurons.

of PFC was more than 12 neurons. The magnitude of noise also affected ghost wave rate. The ghost wave rate reached 80% when the height of PFC was 8 neurons and the noise was 0.015 Hz. When the height of PFC was more than 12 PFC neurons, the ghost wave rates were greater than 100%, which meant there were more than one ghost waves in a trial. Both fail rate and ghost wave rate will affect learning efficiency of PFC→STR synapses (see Section PFC-to-STR Synapses Learn to Predict the Timing of US). Therefore, based on these analyses, we set the height of PFC to 4 neurons and set the noise to 0.01 Hz for stable wave propagation.

PFC-to-STR Synapses Learn to Predict the Timing of Us

With the wave propagation mechanism in place, we still required a means to pair neutral stimuli (e.g., color) with innate value (e.g., a preferred touch). Therefore, we implemented dopamine modulated STDP and wave propagation in the network to associate a CS with a US with precise timing. The main function of the CS input coming from the PFC and STR in the decremental pathway is to predict the timing and strength of the ensuing US and to balance the excitatory and inhibitory forces on the DA neurons. **Figure 7** shows simulation results that explain the underlying neural mechanisms. For each trial in the simulation, CS activated the PFC at 0 ms and triggered a propagating wave of activity. A DA burst was activated at 1100 ms to simulate the effect of US. We ran the simulation for 400 trials. The raster plots and histograms of STR activity for trial 1, trial 100, and trial 400 are shown on the left side of **Figures 7A–C**, respectively. The PFC→STR weights were subject to DA-STDP. The PFC→STR synaptic weights for trial 1, trial 100, and trial 400 are shown on the right side of **Figures 7A–C**, respectively. In trial 1, a burst in DA increases dopamine concentration in STR through dopaminergic projections around 1100 ms. Because the DA-PSF facilitated STR activity, STR neurons were further activated by pre-synaptic PFC neurons around 1100 ms. Since the CS triggered a wave propagating in PFC, there was always a small portion of PFC firing at any moment. Thus, the set of

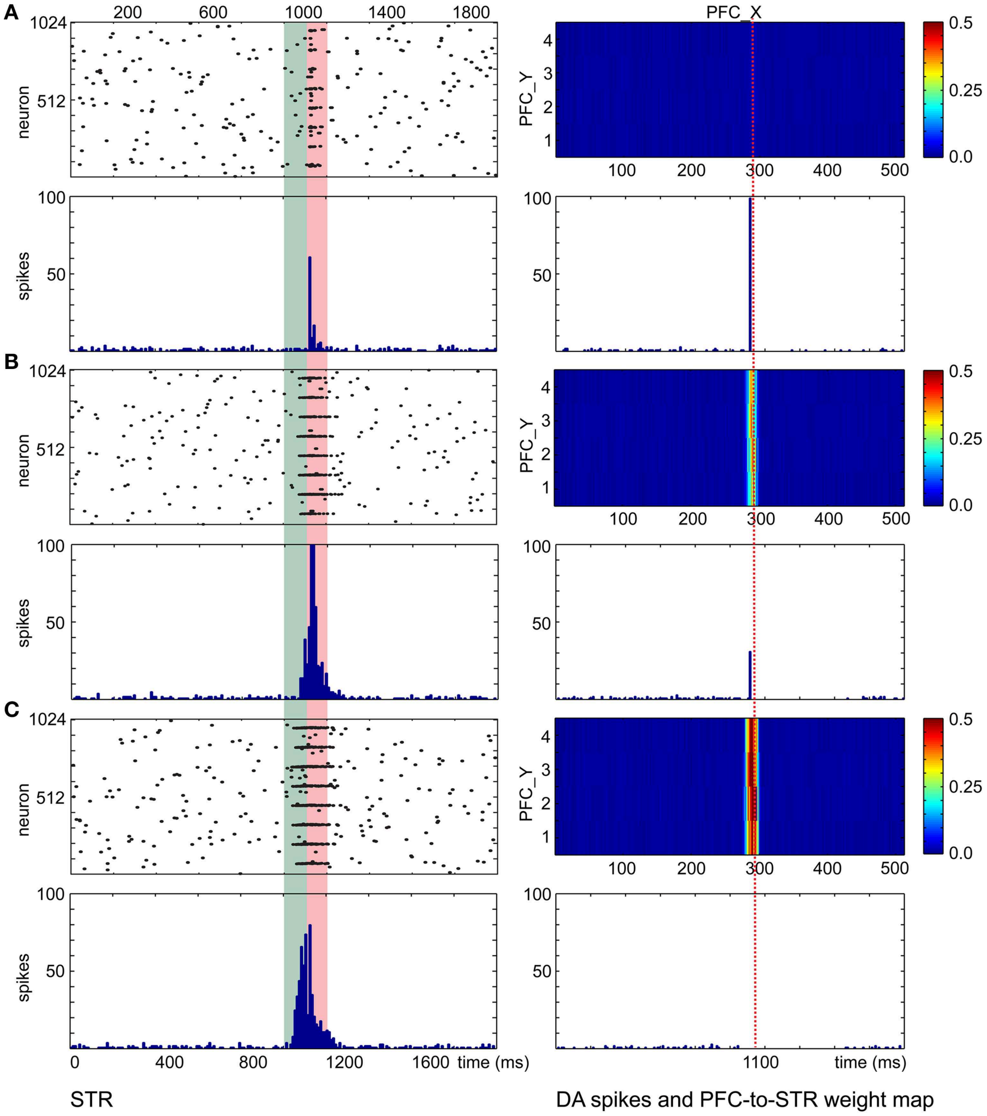


FIGURE 7 | PFC→STR connection learns the timing of dopaminergic bursts. (A) Before learning, a DA burst will trigger transient STR activity through dopamine-modulated post-synaptic facilitation. The wave traveling at PFC area matches the STR activity and PFC→STR synaptic weights are reinforced by DA-STDP. The heatmaps on the right of the figure show the PFC→STR weights at different points, where the color represents the summation of the synaptic weights from a PFC neuron to a STR neuron. For instance, the PFC neuron at (290, 2) has 0.5 as the summation of its PFC→STR synaptic weights **(B)** During learning, PFC→STR synaptic weights get stronger and the traveling wave at PFC area can actively trigger spikes at STR area, leading to a weaker DA burst. **(C)** After learning, PFC→STR synaptic weights reach a maximum strength and the inhibitory force from STR area to DA area prevents a DA burst.

PFC neurons firing around 1100 ms caused their post-synaptic STR neurons to be potentiated. This neural mechanism led to a phasic neural activity around 1120 ms (due to axonal delay and latency of conductance based synapses) as shown in **Figure 7A**. The higher dopamine concentration not only facilitated firings of STR neurons but also boosted the PFC→STR synaptic weights through DA-STDP. This process was repeated trial by trial. Over time, the set of PFC→STR synapses with the appropriate timing were strengthened. Comparing the weight maps in **Figure 7**, a band of strong weights appeared at the PFC neurons encoding 1100 ms (those with x-coordinate from 280 to 300) in **Figure 7B** and got stronger and earlier (i.e., shift toward the left) in **Figure 7C**. To summarize, these strong synapses led to phasic neural activity in STR (see STR raster plots and histograms), which in turn suppressed the DA burst at the precise time through the decremental pathway (see DA raster plots).

CARL-SJR's Behaviors during Learning Multiple CS-US Pairs

The previous sections showed through simulation that the SNN could (1) encode tactile patterns, (2) encode timing through propagating waves, and (3) learn associations between neutral and value-laden stimuli. In this section, we show how these mechanisms can be used in real-world human robot interactive learning experiments.

We conducted the conditioning experiments described in Section Experimental Paradigm with different pairings of color and touch. In the first experiment (see **Figure 8**), we set TH→pIC synaptic weights to 0.04 and CARL-SJR's display alternated between blue, and yellow. The user rubbed CARL-SJR in the downward direction whenever he/she saw the yellow pattern, and in the rightward direction when he/she saw the blue pattern. There were 160 trials for each color pattern, totally 320 trials. We recorded the DA responses for US and CS every trial and sampled the PFC→STR synaptic weights every 40 trials. The colored lines in **Figure 8A** show the 75th percentile, median, and 25th percentile of DA response to the CS over 20 trials (totally 40 trials, 20 trials for each color pattern) while the gray lines show DA response to US. The trends for CS and US are clear; DA response shifted from US to CS for both color patterns as has been observed in empirical studies (Ljungberg et al., 1992; Schultz, 1998; Pan et al., 2005). **Figure 8B** shows the average EXP→INT synaptic weights for each CS (i.e., red, green, blue, and yellow) during conditioning. Since the user only reinforced the blue and yellow patterns, the average synaptic weights directly reflect the user's preferential conditioning as shown by the higher values of blue and yellow EXP→INT weights. The PFC→STR weight maps of trial 40 and trial 320 are shown in **Figure 8C**. These weight maps, which were driven by the user conditioning CARL-SJR, exhibit a strong group of weights associated with the CS with several bands of weights to a lesser degree. The width of a band indicates the imprecise timing of the US and the strength of a band indicates the probability of a US occurrence at the corresponding timing. The DA response to horizontal movements (i.e., US) decreased faster than vertical movements due to the stronger weights for the blue pattern. The weight maps generated by real time tactile inputs also demonstrated our approach can capture the timing and strength of US over a wide range and in a noisy, real-world environment. An interesting behavior is that CARL-SJR slightly prefers touches in the horizontal, rightward direction. In Section Control CARL-SJR's Tactile Preference through Insular Cortex Model, we showed that CARL-SJR by default prefer horizontal movements if we set TH→pIC synaptic weights to 0.04 for all channels. Because of this asymmetry, CARL-SJR shifted DA response to blue pattern faster than yellow pattern (see the higher blue median value in **Figure 8A** and the higher blue line in **Figure 8B**) by the default preference. The result here is also consistent with Section Control CARL-SJR's Tactile Preference through Insular Cortex Model.

In a second set of experiments, we focused on the robot's behavior in the form of conditioned and unconditioned responses. For the conditioned response, which was based on INT activity, CARL-SJR rotated its body. For the unconditioned response, which was based on mIC activity, CARL-SJR emitted a high tone. The US was a downward movement, and the CS was the yellow pattern. We collected data for five runs and each run contained 120 trials. Each data point in **Figure 8D** was calculated as the probability of CR and UR every five trials over five runs. The trends for CR and UR are clear and consistent with the DA spikes triggered by CS and US (see **Figure 8A**). CARL-SJR learned to exhibit the CR with high probability after 40 trials and also suppressed the UR after around 70 trials. Taken together, these human-robot interaction experiments show that the proposed mechanisms can support learning in a real-world, noisy environment.

Extinguishing Behaviors after Learning

The ability to unlearn prior associations is critical for flexible behavior. In conditioning paradigms, omitting the US after learning can lead to extinction of the conditioned response. After conditioning, we repeated the experiment described in Section CARL-SJR's Behaviors during Learning Multiple CS-US Pairs without presenting the US in an additional 200 trials. The DA response and CARL-SJR's behavior were recorded as well. **Figure 9A** shows a representative example for a dip of DA response due to the absence of expected rewards (Ljungberg et al., 1991). The raster plot of DA spikes shows the phasic neural activity around 1100 ms, which was triggered by CS via the incremental pathway. At 1300 ms, the CS arrived at PFC and triggered wave propagation. Based on the weight map developed during learning process, STR neurons exhibited a phasic neural activity around 2000 ms. Since no excitatory force was present around 2000 ms (due to absence of US), the inhibitory force from STR suppressed DA neurons and created a 400 ms interval without DA spikes. In **Figure 9B**, we show the distribution of dip durations in 200 trials. In these experiments, if there was dip in DA activity longer than 400 ms, CARL-SJR emitted a low tone signaling the omission of an expected reward. During this experiment we observed CARL-SJR sang this unhappy low tone in 167 out of 200 trials, which is 83.5%.

To emulate the *in vivo* recordings from Ljungberg's experiments (Ljungberg et al., 1991), we randomly selected 5 out of 100 DA neurons over 200 trials. This emulated recording from a small sample of the available pool of dopaminergic neurons. From these recordings, we composed a raster plot and a histogram in **Figure 9C**. The dip duration is roughly 500 ms (see light red bar in **Figure 9C**) and there is no DA activity for around 100 ms (see deep red bar in **Figure 9C**). These dip durations closely matches those reported in Ljungberg's study.

The DA dips, shown in **Figure 9**, have been suggested to promote extinction when the reward associated with the US is absent. Therefore, we tested extinction behavior in a classical conditioning paradigm (see **Figure 10**). After conditioning, we repeated the experiment without presenting the US. We then recorded the CR and UR for 5 runs and each run had 200 trials. We calculated the probability of the CR every 5 trials over 5 runs. As is indicative of extinction of a behavior, the probability of CR decayed to zero after 50–60 trials (see blue line in **Figure 10**). We also plotted the average EXP→INT synaptic weights for the yellow pattern (see orange line in **Figure 10**). The weights decayed and exhibited fluctuation around a steady level. The decay was due to dips of dopamine (see **Figure 9B**) and noisy spontaneous firing activities in EXP and INT.

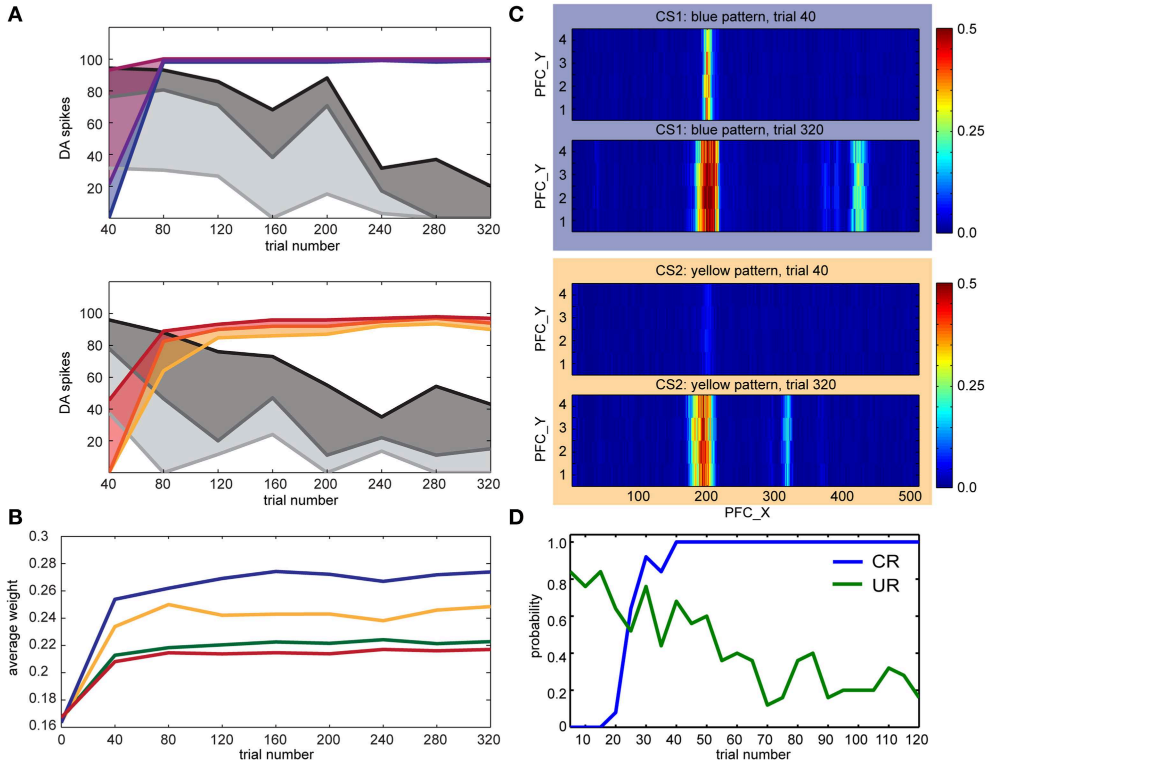


FIGURE 8 | CARL-SJR learned user preferences for color patterns. (A) The colored lines show the 75th percentile, median, and 25th percentile of DA response to the CS over 20 trials (totally 40 trials, 20 trials for each color pattern) while the gray lines show DA response to US. **(B)** The average INT→EXP synaptic weights over 20 trials (totally 40 trials, 20 trials for each color during conditioning. The user only reinforced the blue and yellow patterns. **(C)** The PFC→STR weight maps of blue and yellow patterns over trial 40 and trial 320. The interpretation for the heatmap is the same as **Figure 7**. **(D)** The probability of CR and UR every five trials over five runs. The trends for CR and UR are clear and consistent with the DA spikes triggered by CS and US in **(A)**.

Discussion

We introduced a tactile neurorobot, called CARL-SJR, which was capable of sensing noisy, real-world tactile inputs in a highly uncertain environment and taking tactile inputs with minimal pre-processing to convert touch events into spiking activity (see Section Control CARL-SJR's Tactile Preference through Insular Cortex Model). A detailed spiking neural network (SNN) model of somatosensory cortex and the insular cortex, which is known to be important for hedonic touch, drove CARL-SJR's behavior. Learning in the model was driven by a dual pathway model of dopaminergic learning and the emergence of traveling waves of neural activity that governed the release of dopamine and the timing of CS and US (see Sections Wave Propagation in PFC and PFC-to-STR Synapses Learn to Predict the Timing of US). CARL-SJR demonstrated the ability to associate multiple CS's (i.e., color patterns displayed on its shell) with US's (i.e., user hand sweeps across its shell). For example, in Section CARL-SJR's Behaviors during Learning Multiple CS-US Pairs, we showed that the model supported associations between blue-right and yellow-down during a training session. Moreover, the model was able to learn despite trial-by-trial variations in CS-US intervals due to the uncertainties of user inputs.

The human-robot interaction studies with CARL-SJR exhibits the paradigm of mutual reinforced learning. Specifically, the robot can learn the user's preferences through conditioning tasks while the user can learn the robot's tactile preference through the robot's responses. CARL-SJR's style of human-robot interaction may be applications in socially assistive robotics (Scassellati et al., 2012) or socially affective robots (Breazeal, 2009). Compared to other social robots, CARL-SJR is somewhat unique in that it focuses on tactile rather than visual interaction.

CARL-SJR was designed to encourage interaction through touch. Tactile sensing is an active area of robotics research that takes inspiration from biology and neuroscience. For example, inspired by rodents and other mammals with vibrissae, whiskered robots have been developed to sense the borders and shape of objects (N'Guyen et al., 2010; Pearson et al., 2011; Evans et al., 2012; Schroeder and Hartmann, 2012). Fingers and hands have been developed for humanoid robots to enable grasping and detecting surfaces (Bologna et al., 2011, 2013; Spigler et al., 2012; Li et al., 2013). Most of these robots are constructed from custom-made materials and sensing circuits for touch (Sewards and Sewards, 2002; Cannata et al., 2008; Maheshwari and Saraf, 2008; Dahiya et al., 2010). Creating an artificial tactile system is difficult for many reasons. For example, the sensors must cover a

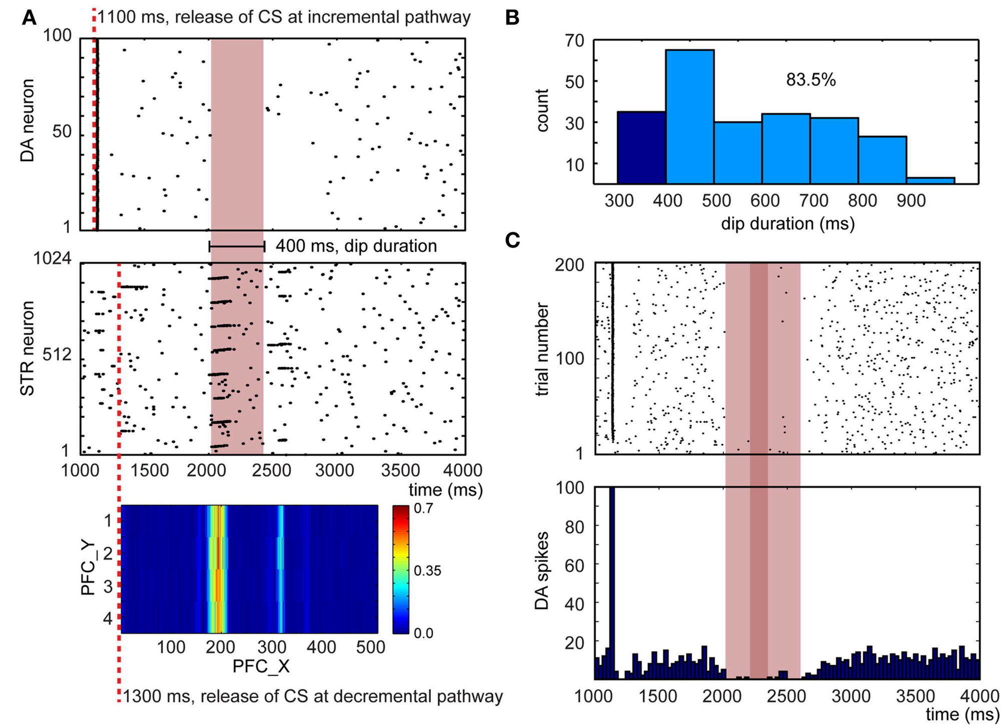


FIGURE 9 | Network response after learning when the US is omitted. **(A)** A representative example for a dip in DA response due to the absence of expected rewards. The heatmap at the bottom shows the PFC→STR weights, where the color represents the summation of the synaptic weights from a PFC neuron to a STR neuron. **(B)** Distribution of dip durations in 200 trials. The threshold for triggering a low tone is set to 400 ms. 83.5% trials trigger a low tone. **(C)** The raster plot and histogram of 5 randomly selected DA neurons over 200 trials when the US is omitted.

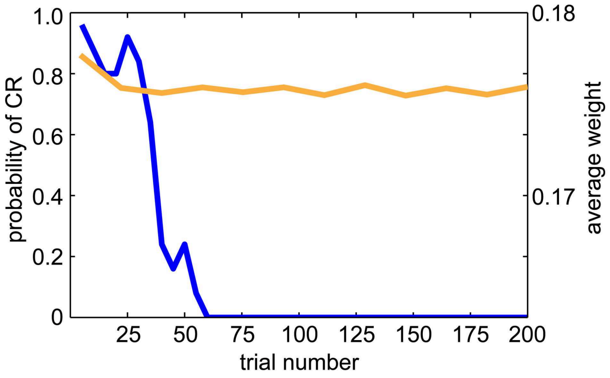


FIGURE 10 | Extinction behavior when the US is omitted. The blue line shows the average CR probability every 5 trials over 5 runs during the extinction trials. The probability of CR decayed to 0.0 after 50 trials, which is consistent with animal studies. The orange line shows the average EXP→INT synaptic weights for the reinforced color pattern. Note that the weights decay to a baseline level during extinction.

large range and be compliant with the surfaces with which they interact. Moreover, the spatiotemporal nature of tactile stimuli, as well as the noisy sensors and environments in which they operate, make the perception of touch a complex problem. To provide a large surface that could handle a wide range of user inputs, we utilized a matrix of trackballs, which are found in many cellphones, across CARL-SJR's curved shell. The size, shape, and resolution was a good fit for the types of social interactions for which CARL-SJR was designed (Bucci et al., 2014).

Driven by the dual-pathway model, the acquisition and extinction behaviors of CARL-SJR are consistent with animal behavioral studies (Rescorla, 1988). Further, the neural activities in our model are consistent with *in vivo* neural recordings as well. First, the robot has built-in tactile preferences. This matches the animals' behaviors wherein they prefer social touches in particular areas and directions. The neural mechanism for integrating tactile information in pIC area could serve as an explanation to animals' innate tactile preferences. Second, assuming the user knows the tactile preferences in advance or learns them, the user can try to reward the robot by touching the robot in its preferred ways. In the present experiments, the unconditioned response (UR) was mapped to mIC activity resulting in a high tone signaled by the robot. The high tone was used as feedback for the user to know that CARL-SJR enjoyed this touch. Third, the robot can learn the association between a color and a gentle touch. In the present paper, a color displaying on the robot's surface, which was spontaneously generated, was considered the conditioned stimulus (CS), and a gentle touch in an innate preferred direction (i.e., unexpected reward) served as the unconditioned stimulus (US). After learning, the robot associated the reinforced color, which was facilitated by DA, with the future reward, which is a preferred touch. Moreover, the DA response shifted from US to CS, as has been observed experimentally (Ljungberg et al., 1992; Schultz, 1998; Pan et al., 2005). CARL-SJR expressed its conditioned response (CR) when INT activity was high and signaled the CR with a spinning motion and bright colors displayed on its surface. Fourth, the robot was depressed if a gentle touch was expected, but not given. This occurred as a dip in DA activity when the expected US was withheld. We mapped the duration of a dip to a low tone, which sounded unhappy. This behavior was triggered by a dip of dopamine concentration level (Ljungberg et al., 1991).

Comparison with Previous Computational Models

To associate temporally separated events (i.e., CS and US), many computational models assumed the firing activities of neurons responding to an earlier event slowly decay and the sustained firing activities are associated neurons with the ensuing US event through STDP (Gluck and Thompson, 1987; Drew and Abbott, 2006). Similarly, the slowly decaying eligibility trace (Houk et al., 1995) has been applied to associate temporally separated events in dopamine modulated STDP (Izhikevich, 2007). The idea of slowly decaying eligibility trace was also successfully applied to rate-based neurons where Soltoggio and Steil used rare neural correlations to calculate the eligibility trace (Soltoggio and Steil, 2013). This approach was further validated on iCub for classical and operant conditioning tasks (Soltoggio et al., 2013). Chorley and Seth later integrated the DA-STDP mechanism into the dual-pathway model (Chorley and Seth, 2011). Their model successfully accounted for a wide range of reward-related DA responses. A different approach to conditioning paradigms is to incorporate temporally separated events as propagating spiking waves and associate these events through the spatiotemporal interaction of these waves (Palmer and Gong, 2014).

Using wave propagation to solve the temporal credit assignment problem has some interesting features that address limitations in other neurobiologically plausible reinforcement learning rules. The early dopamine model of reinforcement learning was very similar to temporal difference learning (Montague et al., 1996; Schultz et al., 1997). From empirical data and computational modeling, dopamine appeared to track the reward prediction error. However, in the model, the dopamine signal moved backward in time in successive trials until it corresponded to the stimulus that was predictive of reward. This movement of a dopamine signal over time has not been observed empirically. Others have proposed an eligibility trace as a means to solve the credit assignment problem (Izhikevich, 2007; Soltoggio and Steil, 2013). However, since the amplitude of the trace from the time of the CS to the time of the US can be quite small, it requires many trials to make a strong association. Moreover, there is little empirical support for a biological process to support this type of learning that lasts over many seconds. In contrast, wave propagation has empirical support and does not have the limitations described above (Rubino et al., 2006; Benucci et al., 2007; Ferezou et al., 2007; Han et al., 2008; Wu et al., 2008; Lubenov and Siapas, 2009; Sato et al., 2012). These waves have the appropriate timescale, are robust to variations in timing, and can send a strong enough signal to support associative learning in a plausible number of trials.

Another interesting approach to solving the credit assignment was proposed by Khamassi et al. (2011), where dopamine neurons produced a reward prediction error signal in response to any salient event and affected synaptic plasticity when it co-occurred with a motor efference property. This would also address the limitations described above. However, in our present experiments we do not have a motor efference copy. The CS is a color display on the robot's shell and does not produce a motor action. The motor command is actually from the subject interacting with the robot. Still, this may be an interesting approach to implement in future models.

CARL-SJR's SNN model, which integrated the slow C-fiber and fast Aβ-fiber tactile pathways to the pIC, Chorley and Seth's dual-pathway model, and Palmer's spiking wave propagation, demonstrated associative learning in the real-world with a robot receiving noisy user input. Several prerequisites or neural behaviors in Chorley's and Palmer's work limit their real-world applications. Palmer's model successfully shifts the neural response from US to CS. However, the US could trigger both a CR and a UR at the same time if the CS is not presented before US. PFC activity in Chorley and Seth's model was implemented with a pre-generated polychronous group (Izhikevich, 2006). The appearance of consistent polychronous groups in a noisy environment is difficult. For example, the criteria for re-occurrence of a polychronous group was set to a low threshold (i.e., 25% neurons of a group) in Szatmary and Izhikevich (2010). In this case, only a small portion of neurons showed time-locked spike patterns. In contrast, when the criteria for re-occurrence was set to a high threshold (e.g., 100% neurons of a group) as in Bucci et al. (2014), the polychronous groups were very small, rarely occurred, and lasted for less than 40 ms. In this prior neurorobot study, these polychronous groups did not match the PFC activity in Chorley's model where time-locked cortical spike patterns sustained for 1 s. Both Chorley's and our model exhibit an interesting neural mechanism in which the STR response was a little bit later than the DA response in early trials (see **Figure 7A**) and then shifted backward to match the timing of the DA response in late trials (see **Figure 7C**). Whereas Chorley and Seth did not discuss how the PFC→STR synaptic weights may affect the learning progress, we showed the PFC→STR weight maps were indicative of the US timing and this may support temporal associations.

Our present PFC model incorporated the idea of spiking wave propagation in Palmer's model, which made pre-generated spikes or pre-processing of the CS unnecessary. The location of a neuron in PFC and the topographical projections to STR encoded the time relative to the CS in the decremental pathway. The PFC→STR weight map clearly reflected the learning status (see **Figure 7**). The firing rate of our PFC model is consistent with experimental studies: <0.5 Hz in the resting state (Koch and Fuster, 1989) and 5–40 Hz when behaving (Funahashi et al., 1989). Further, our model has been validated to handle the uncertainty of CS-US interval within 2 s while Chorley reported the valid US window to be (500 ± 100 ms). We also used inhibitory fast spiking neurons to simulate striatal medium spiny neurons (Humphries et al., 2009). The characteristic short spiking burst of a FS neuron facilitated learning in the decremental pathway. The FS neuron transitioned to an excitable state due to DA-PSF, and activity from a pre-synaptic RS neuron in PFC triggered a spike train in the post-synaptic FS STR neuron. This resulted in multiple increases in the synaptic weights through LTP. This scenario was dependent on increased dopamine activity.

The model guiding CARL-SJR's behavior was also unique in that it implemented the separate and parallel pathways for transmitting touch information to the cortex, in which fine touch is well represented in the somatosensory cortex, and hedonic caressing appears to represented in the insular cortex (Sewards and Sewards, 2002; Olausson et al., 2010; Morrison et al., 2011a). The response of the fast Aβ-fiber tactile pathway is reflected in the fine resolution somatotopic response shown in **Figure 3A**. Less is known about insular neuronal activity in response to touch. However, it has been suggested that the insular cortex can make predictions of internal states (Singer et al., 2009; Seth, 2013) and the neural architecture driving CARL-SJR's pleasure seeking behavior supports these claims (see **Figure 2**).

CARL-SJR's Limitations

CARL-SJR has several limitations. We developed a PFC wave with relatively low internal noise (i.e., spontaneous neural activity). The maximum noise in **Figure 6B** is 0.015 Hz, which is 10 times smaller than the spontaneous firing activity (i.e., white noise) in resting PFC suggested by the experimental study of Koch and Fuster (1989). The low spontaneous firing activity extremely slows down the extinction process in the conditioning task (see Section Extinguishing Behaviors after Learning) because the probability to decorrelate coupled neurons in PFC and STR is too low. We left this issue to a future improvement on developing wave propagation under a typical noise level.

The peak dopamine value in our model is 20 μM, which is much higher than 3 μM reported in Izhikevich's and Chorley's models. The range of dopamine value was tuned to result in adequate learning rates in the neurorobot experiments. We could lower the dopamine value and keep the observable learning speed if some compensatory neural mechanisms were implemented. For example, a replay mechanism for the CS and US pairings when CARL-SJR "sleeps" (Buzsaki, 1998). However, to incorporate a true replay mechanism would require substantial efforts in building a hippocampus model for offline learning (Khamassi and Humphries, 2012). A simple approach, which would be effective in the present architecture, would be to simulate replay by injecting the CS and US into the model when CARL-SJR is not actively behaving.

The neural response of mechanoreceptors with C-fiber is tuned for gentle speeds (Morrison et al., 2011a). We did not capture this characteristic because the trackballs, as they are currently designed, cannot detect speed locally (i.e., the number of touch events is not proportional to the rolling speed of a trackball). We could calculate the speed of a movement across multiple trackballs. However, this approach requires substantial pre-processing and therefore, contradicts our design choice.

Conclusions

CARL-SJR is a neurorobot whose behavior is guided by a SNN model of tactile pathways in the cortex, and demonstrates user defined entraining of a robot through touch. By incorporating dopamine modulated learning with traveling waves of neural activity, we have shown a biologically plausible method of instrumental conditioning. Our SNN model can be easily extended for robotic applications in reinforcement learning paradigms. Future directions include dopaminergic projections to the frontal cortex and implementation of the serotoninergic (5-HT) system (Krichmar, 2013). DA is linked to rewards and curiosity-seeking behavior and 5-HT may be linked to risk aversion and withdrawn behavior (Tops et al., 2009; Boureau and Dayan, 2011; Siegel and Crockett, 2013). It has also been suggested that 5-HT plays a role in waiting for a delayed reward (Miyazaki et al., 2012). By combining the DA and 5-HT systems, we could make CARL-SJR not only learn through rewards but also cost. Moreover, following the notion of 5-HT being related temporal discounting, we could use 5-HT levels to modulate CARL-SJR's impulsiveness. These additions would make CARL-SJR's behavior more interesting and such a system could have applications in the fields of socially assistive and socially affective robotics.

Acknowledgments

This research was supported by NSF under Grant IIS-0910710 and by a subcontract (THE-207651) from Qualcomm Technologies Incorporated.

Supplementary Material

The Supplementary Material for this article can be found online at: http://journal.frontiersin.org/article/10.3389/fnbot.2015.00006

References

Abraira, V. E., and Ginty, D. D. (2013). The sensory neurons of touch. *Neuron* 79, 618–639. doi: 10.1016/j.neuron.2013.07.051

Benucci, A., Frazor, R. A., and Carandini, M. (2007). Standing waves and traveling waves distinguish two circuits in visual cortex. *Neuron* 55, 103–117. doi: 10.1016/j.neuron.2007.06.017

Bologna, L. L., Pinoteau, J., Brasselet, R., Maggiali, M., and Arleo, A. (2011). Encoding/decoding of first and second order tactile afferents in a neurorobotic application. *J. Physiol. Paris* 105, 25–35. doi: 10.1016/j.jphysparis.2011.08.002

Bologna, L. L., Pinoteau, J., Passot, J. B., Garrido, J. A., Vogel, J., Vidal, E. R., et al. (2013). A closed-loop neurobotic system for fine touch sensing. *J. Neural Eng.* 10:046019. doi: 10.1088/1741-2560/10/4/046019

Boureau, Y. L., and Dayan, P. (2011). Opponency revisited: competition and cooperation between dopamine and serotonin. *Neuropsychopharmacology* 36, 74–97. doi: 10.1038/npp.2010.151

Breazeal, C. (2009). Role of expressive behaviour for robots that learn from people. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 364, 3527–3538. doi: 10.1098/rstb.2009.0157

Brown, J., Bullock, D., and Grossberg, S. (1999). How the basal ganglia use parallel excitatory and inhibitory learning pathways to selectively respond to unexpected rewarding cues. *J. Neurosci.* 19, 10502–10511.

Bucci, L. D., Chou, T.-S., and Krichmar, J. L. (2014). "Sensory decoding in a tactile, interactive neurorobot," in *Paper Presented at the Robotics and Automation (ICRA), 2014 IEEE International Conference on* (Hong Kong).

Buzsaki, G. (1998). Memory consolidation during sleep: a neurophysiological perspective. *J. Sleep Res.* 7(Suppl. 1), 17–23.

Cannata, G., Maggiali, M., Metta, G., and Sandini, G. (2008). "An embedded artificial skin for humanoid robots," in *Paper Presented at the Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on* (Seoul).

Caporale, N., and Dan, Y. (2008). Spike timing-dependent plasticity: a Hebbian learning rule. *Annu. Rev. Neurosci.* 31, 25–46. doi: 10.1146/annurev.neuro.31.060407.125639

Carlson, K. D., Nageswaran, J. M., Dutt, N., and Krichmar, J. L. (2014). An efficient automated parameter tuning framework for spiking neural networks. *Front. Neurosci.* 8:10. doi: 10.3389/fnins.2014.00010

Chen, Y., McKinstry, J. L., and Edelman, G. M. (2013). Versatile networks of simulated spiking neurons displaying winner-take-all behavior. *Front. Comput. Neurosci.* 7:16. doi: 10.3389/fncom.2013.00016

Chorley, P., and Seth, A. K. (2011). Dopamine-signaled reward predictions generated by competitive excitation and inhibition in a spiking neural network model. *Front. Comput. Neurosci.* 5:21. doi: 10.3389/fncom.2011.00021

Craig, A. D. (2002). How do you feel? Interoception: the sense of the physiological condition of the body. *Nat. Rev. Neurosci.* 3, 655–666. doi: 10.1038/nrn894

Craig, A. D. (2009). How do you feel—now? The anterior insula and human awareness. *Nat. Rev. Neurosci.* 10, 59–70. doi: 10.1038/nrn2555

Craig, A. D., Bushnell, M. C., Zhang, E. T., and Blomqvist, A. (1994). A thalamic nucleus specific for pain and temperature sensation. *Nature* 372, 770–773. doi: 10.1038/372770a0

Dahiya, R. S., Metta, G., Valle, M., and Sandini, G. (2010). Tactile sensing—from humans to humanoids. *Robotics IEEE Trans.* 26, 1–20. doi: 10.1109/TRO.2009.2033627

Drew, P. J., and Abbott, L. F. (2006). Extending the effects of spike-timing-dependent plasticity to behavioral timescales. *Proc. Natl. Acad. Sci. U.S.A.* 103, 8876–8881. doi: 10.1073/pnas.0600676103

Evans, M. H., Fox, C. W., Lepora, N. F., Pearson, M. J., Sullivan, J. C., and Prescott, T. J. (2012). The effect of whisker movement on radial distance estimation: a case study in comparative robotics. *Front. Neurorobot.* 6:12. doi: 10.3389/fnbot.2012.00012

Felleman, D. J., and Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cereb. Cortex* 1, 1–47.

Ferezou, I., Haiss, F., Gentet, L. J., Aronoff, R., Weber, B., and Petersen, C. C. (2007). Spatiotemporal dynamics of cortical sensorimotor integration in behaving mice. *Neuron* 56, 907–923. doi: 10.1016/j.neuron.2007.10.007

Funahashi, S., Bruce, C. J., and Goldman-Rakic, P. S. (1989). Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *J. Neurophysiol.* 61, 331–349.

Gluck, M. A., and Thompson, R. F. (1987). Modeling the neural substrates of associative learning and memory: a computational approach. *Psychol. Rev.* 94, 176–191.

Griffith, J. S., and Horn, G. (1966). An analysis of spontaneous impulse activity of units in the striate cortex of unrestrained cats. *J. Physiol.* 186, 516–534.

Han, F., Caporale, N., and Dan, Y. (2008). Reverberation of recent visual experience in spontaneous cortical waves. *Neuron* 60, 321–327. doi: 10.1016/j.neuron.2008.08.026

Houk, J. C., Adams, J. L., and Barto, A. G. (1995). "A model of how the basal ganglia generate and use neural signals that predict reinforcement," in *Models of Information Processing in the Basal Ganglia*, eds J. C. Houk, J. L. Davis, and D. G. Beiser (Cambridge, MA: MIT Press), 249–270.

Humphries, M. D., Lepora, N., Wood, R., and Gurney, K. (2009). Capturing dopaminergic modulation and bimodal membrane behaviour of striatal medium spiny neurons in accurate, reduced models. *Front. Comput. Neurosci.* 3:26. doi: 10.3389/neuro.10.026.2009

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Izhikevich, E. M. (2006). Polychronization: computation with spikes. *Neural Comput.* 18, 245–282. doi: 10.1162/089976606775093882

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cereb. Cortex* 17, 2443–2452. doi: 10.1093/cercor/bhl152

Izhikevich, E. M., and Edelman, G. M. (2008). Large-scale model of mammalian thalamocortical systems. *Proc. Natl. Acad. Sci. U.S.A.* 105, 3593–3598. doi: 10.1073/pnas.0712231105

Khamassi, M., and Humphries, M. D. (2012). Integrating cortico-limbic-basal ganglia architectures for learning model-based and model-free navigation strategies. *Front. Behav. Neurosci.* 6:79. doi: 10.3389/fnbeh.2012.00079

Khamassi, M., Lallee, S., Enel, P., Procyk, E., and Dominey, P. F. (2011). Robot cognitive control with a neurophysiologically inspired reinforcement learning model. *Front. Neurorobot.* 5:1. doi: 10.3389/fnbot.2011.00001

Koch, K. W., and Fuster, J. M. (1989). Unit activity in monkey parietal cortex related to haptic perception and temporary memory. *Exp. Brain Res.* 76, 292–306.

Krichmar, J. L. (2013). A neurorobotic platform to test the influence of neuromodulatory signaling on anxious and curious behavior. *Front. Neurorobot.* 7:1. doi: 10.3389/fnbot.2013.00001

Krichmar, J. L., and Rohrbein, F. (2013). Value and reward based learning in neurorobots. *Front. Neurorobot.* 7:13. doi: 10.3389/fnbot.2013.00013

Li, Q., Elbrechter, C., Haschke, R., and Ritter, H. (2013). "Integrating vision, haptics and proprioception into a feedback controller for in-hand manipulation of unknown objects," in *Paper Presented at the Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on* (Tokyo).

Ljungberg, T., Apicella, P., and Schultz, W. (1991). Responses of monkey midbrain dopamine neurons during delayed alternation performance. *Brain Res.* 567, 337–341.

Ljungberg, T., Apicella, P., and Schultz, W. (1992). Responses of monkey dopamine neurons during learning of behavioral reactions. *J. Neurophysiol.* 67, 145–163.

Lubenov, E. V., and Siapas, A. G. (2009). Hippocampal theta oscillations are travelling waves. *Nature* 459, 534–539. doi: 10.1038/nature08010

Maheshwari, V., and Saraf, R. (2008). Tactile devices to sense touch on a par with a human finger. *Ang. Chem. Intern. Edn.* 47, 7808–7826. doi: 10.1002/anie.200703693

Markram, H., Gerstner, W., and Sjostrom, P. J. (2011). A history of spike-timing-dependent plasticity. *Front. Synaptic Neurosci.* 3:4. doi: 10.3389/fnsyn.2011.00004

Miyazaki, K. W., Miyazaki, K., and Doya, K. (2012). Activation of dorsal raphe serotonin neurons is necessary for waiting for delayed rewards. *J. Neurosci.* 32, 10451–10457. doi: 10.1523/JNEUROSCI.0915-12.2012

Montague, P. R., Dayan, P., and Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *J. Neurosci.* 16, 1936–1947.

Morrison, I., Bjornsdotter, M., and Olausson, H. (2011a). Vicarious responses to social touch in posterior insular cortex are tuned to pleasant caressing speeds. *J. Neurosci.* 31, 9554–9562. doi: 10.1523/JNEUROSCI.0397-11.2011

Morrison, I., Loken, L. S., Minde, J., Wessberg, J., Perini, I., Nennesmo, I., et al. (2011b). Reduced C-afferent fibre density affects perceived pleasantness and empathy for touch. *Brain* 134(Pt 4), 1116–1126. doi: 10.1093/brain/awr011

N'Guyen, S., Pirim, P., and Meyer, J.-A. (2010). "Tactile texture discrimination in the robot-rat psikharpax," in *Paper Presented at the BIOSIGNALS* (Valencia).

Nageswaran, J. M., Dutt, N., Krichmar, J. L., Nicolau, A., and Veidenbaum, A. V. (2009). A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. *Neural Netw.* 22, 791–800. doi: 10.1016/j.neunet.2009.06.028

Nicola, S. M., Surmeier, J., and Malenka, R. C. (2000). Dopaminergic modulation of neuronal excitability in the striatum and nucleus accumbens. *Annu. Rev. Neurosci.* 23, 185–215. doi: 10.1146/annurev.neuro.23.1.185

Olausson, H., Wessberg, J., Morrison, I., McGlone, F., and Vallbo, A. (2010). The neurophysiology of unmyelinated tactile afferents. *Neurosci. Biobehav. Rev.* 34, 185–191. doi: 10.1016/j.neubiorev.2008.09.011

Palmer, J. H., and Gong, P. (2014). Associative learning of classical conditioning as an emergent property of spatially extended spiking neural circuits with synaptic plasticity. *Front. Comput. Neurosci.* 8:79. doi: 10.3389/fncom.2014.00079

Pan, W. X., Schmidt, R., Wickens, J. R., and Hyland, B. I. (2005). Dopamine cells respond to predicted events during classical conditioning: evidence for eligibility traces in the reward-learning network. *J. Neurosci.* 25, 6235–6242. doi: 10.1523/JNEUROSCI.1478-05.2005

Pearson, M. J., Mitchinson, B., Sullivan, J. C., Pipe, A. G., and Prescott, T. J. (2011). Biomimetic vibrissal sensing for robots. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 366, 3085–3096. doi: 10.1098/rstb.2011.0164

Rescorla, R. A. (1988). Behavioral studies of Pavlovian conditioning. *Annu. Rev. Neurosci.* 11, 329–352. doi: 10.1146/annurev.ne.11.030188.001553

Richert, M., Nageswaran, J. M., Dutt, N., and Krichmar, J. L. (2011). An efficient simulation environment for modeling large-scale cortical processing. *Front. Neuroinform.* 5:19. doi: 10.3389/fninf.2011.00019

Rubino, D., Robbins, K. A., and Hatsopoulos, N. G. (2006). Propagating waves mediate information transfer in the motor cortex. *Nat. Neurosci.* 9, 1549–1557. doi: 10.1038/nn1802

Sato, T. K., Nauhaus, I., and Carandini, M. (2012). Traveling waves in visual cortex. *Neuron* 75, 218–229. doi: 10.1016/j.neuron.2012.06.029

Scassellati, B., Admoni, H., and Mataric, M. (2012). Robots for use in autism research. *Annu. Rev. Biomed. Eng.* 14, 275–294. doi: 10.1146/annurev-bioeng-071811-150036

Schroeder, C. L., and Hartmann, M. J. (2012). Sensory prediction on a whiskered robot: a tactile analogy to "optical flow". *Front. Neurorobot.* 6:9. doi: 10.3389/fnbot.2012.00009

Schultz, W. (1998). Predictive reward signal of dopamine neurons. *J. Neurophysiol.* 80, 1–27.

Schultz, W. (2006). Behavioral theories and the neurophysiology of reward. *Annu. Rev. Psychol.* 57, 87–115. doi: 10.1146/annurev.psych.56.091103.070229

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science* 275, 1593–1599.

Seth, A. K. (2013). Interoceptive inference, emotion, and the embodied self. *Trends Cogn. Sci.* 17, 565–573. doi: 10.1016/j.tics.2013.09.007

Sewards, T. V., and Sewards, M. (2002). Separate, parallel sensory and hedonic pathways in the mammalian somatosensory system. *Brain Res. Bull.* 58, 243–260. doi: 10.1016/S0361-9230(02)00783-9

Siegel, J. Z., and Crockett, M. J. (2013). How serotonin shapes moral judgment and behavior. *Ann. N.Y. Acad. Sci.* 1299, 42–51. doi: 10.1111/nyas.12229

Silver, R. A. (2010). Neuronal arithmetic. *Nat. Rev. Neurosci.* 11, 474–489. doi: 10.1038/nrn2864

Singer, T., Critchley, H. D., and Preuschoff, K. (2009). A common role of insula in feelings, empathy and uncertainty. *Trends Cogn. Sci.* 13, 334–340. doi: 10.1016/j.tics.2009.05.001

Soltoggio, A., Lemme, A., Reinhart, F., and Steil, J. J. (2013). Rare neural correlations implement robotic conditioning with delayed rewards and disturbances. *Front. Neurorobot.* 7:6. doi: 10.3389/fnbot.2013.00006

Soltoggio, A., and Steil, J. J. (2013). Solving the distal reward problem with rare correlations. *Neural Comput.* 25, 940–978. doi: 10.1162/NECO_a_00419

Spigler, G., Oddo, C. M., and Carrozza, M. C. (2012). "Soft-neuromorphic artificial touch for applications in neuro-robotics," in *Paper Presented at the Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on* (Rome).

Srinivasa, N., and Jiang, Q. (2013). Stable learning of functional maps in self-organizing spiking neural networks with continuous synaptic plasticity. *Front. Comput. Neurosci.* 7:10. doi: 10.3389/fncom.2013.00010

Szatmary, B., and Izhikevich, E. M. (2010). Spike-timing theory of working memory. *PLoS Comput. Biol.* 6:879. doi: 10.1371/journal.pcbi.1000879

Tan, C. O., and Bullock, D. (2008). A local circuit model of learned striatal and dopamine cell responses under probabilistic schedules of reward. *J. Neurosci.* 28, 10062–10074. doi: 10.1523/JNEUROSCI.0259-08.2008

Tops, M., Russo, S., Boksem, M. A., and Tucker, D. M. (2009). Serotonin: modulator of a drive to withdraw. *Brain Cogn.* 71, 427–436. doi: 10.1016/j.bandc.2009.03.009

Williams, G. V., and Castner, S. A. (2006). Under the curve: critical issues for elucidating D1 receptor function in working memory. *Neuroscience* 139, 263–276. doi: 10.1016/j.neuroscience.2005.09.028

Wu, J. Y., Xiaoying, H., and Chuan, Z. (2008). Propagating waves of activity in the neocortex: what they are, what they do. *Neuroscientist* 14, 487–502. doi: 10.1177/1073858408317066

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: 23 July 2015
doi: 10.3389/fnbot.2015.00007

A neural network-based exploratory learning and motor planning system for co-robots

Byron V. Galbraith[1, 2, 3]*, *Frank H. Guenther*[2, 4, 5] and *Massimiliano Versace*[2, 3]

[1] Program in Cognitive and Neural Systems, Boston University, Boston, MA, USA, [2] Center for Excellence in Learning in Education, Science, and Technology, Boston University, Boston, MA, USA, [3] Neuromorphics Laboratory, Boston University, Boston, MA, USA, [4] Department of Speech, Language, and Hearing Sciences, Boston University, Boston, MA, USA, [5] Department of Biomedical Engineering, Boston University, Boston, MA, USA

Collaborative robots, or co-robots, are semi-autonomous robotic agents designed to work alongside humans in shared workspaces. To be effective, co-robots require the ability to respond and adapt to dynamic scenarios encountered in natural environments. One way to achieve this is through exploratory learning, or "learning by doing," an unsupervised method in which co-robots are able to build an internal model for motor planning and coordination based on real-time sensory inputs. In this paper, we present an adaptive neural network-based system for co-robot control that employs exploratory learning to achieve the coordinated motor planning needed to navigate toward, reach for, and grasp distant objects. To validate this system we used the 11-degrees-of-freedom RoPro Calliope mobile robot. Through motor babbling of its wheels and arm, the Calliope learned how to relate visual and proprioceptive information to achieve hand-eye-body coordination. By continually evaluating sensory inputs and externally provided goal directives, the Calliope was then able to autonomously select the appropriate wheel and joint velocities needed to perform its assigned task, such as following a moving target or retrieving an indicated object.

Keywords: co-robot, exploratory learning, motor planning, neural network, egocentric navigation, embodied AI

Edited by:
Andrea Soltoggio,
Loughborough University, UK

Reviewed by:
Mostafa Ajallooeian,
École Polytechnique Fédérale de Lausanne, Switzerland
Andre Lemme,
Bielefeld Universitaet, Germany

Correspondence:
Byron V. Galbraith, Program in Cognitive and Neural Systems, Boston University, 677 Beacon St, Boston, MA 02215, USA
byron.galbraith@gmail.com

Received: *23 March 2015*
Accepted: *08 July 2015*
Published: *23 July 2015*

Citation:
Galbraith BV, Guenther FH and Versace M (2015) A neural network-based exploratory learning and motor planning system for co-robots. Front. Neurorobot. 9:7. doi: 10.3389/fnbot.2015.00007

Introduction

Co-robots, collaborative robots that work alongside humans to perform assistive tasks, are becoming more prevalent, notably in the healthcare and telepresence spaces (Kristoffersson et al., 2013). A major challenge for co-robots is the need to make decisions on how to operate in dynamic environments with other autonomous agents (Hayes and Scassellati, 2013). This includes using onboard sensors to detect and avoid obstacles or finding, reaching for, and grasping objects. Embodying the co-robot with some sense of spatial awareness is critical for it to make appropriate decisions on how to proceed with its tasks.

Spatial awareness here refers to the combination of sensory inputs, such as visual and proprioceptive, to construct an egocentric coordinate system for objects in the immediate vicinity of the co-robot. The sensory processing, decision-making, and motor planning components of the task process all share this reference frame in order to achieve effective coordination. For instance, the co-robot needs to know where its body and arm are relative to a visually identified target object in order to plan and execute the appropriate motor actions needed to achieve its goal of grasping the object. If the robot is too far away to reach for a target object from its current position, it will have to move its body closer until the target is within range.

A common first step in developing co-robot control models is to employ simulations and virtual environments to evaluate which strategies and methods have a chance of working in the real world. By avoiding issues such as battery charge and wear and tear of robot parts in simulations, multiple models can be evaluated rapidly without fear of damage to physical components. The main drawback to relying on virtual environments is that many challenges faced in the real world are difficult to simulate accurately without significant effort. Perfectly aligned idealized components of a robotic limb in the virtual environment will have isotropic movement behavior, while in the real world, compliance in the mounting joint and inconsistent servo performance will result in anisotropic movements. Even more challenging is the reliance on data from actual sensors, which are susceptible to noise and artifacts, whereas simulated models frequently use perfect information and highly constrained environments.

These variances between idealized models and physical reality may not be describable analytically, which poses a significant challenge in translating theoretical control systems to practical application. One solution is to embody the co-robot with an adaptive system that integrates and learns actual sensory and behavioral data. By using exploratory learning methods, the robotic agent is able to use a form of unsupervised learning where it gains an operational model of its capabilities by observing the results of its own actions. As the co-robot performs and observes the results of endogenous random movements, i.e., motor babbling, it learns how to link sensory information with motor actions. Once these causal relationship models are built, the co-robot can then transition from passively observing undirected actions to actively planning goal-directed actions.

In this work we present such a system using an adaptive neural network-based controller that employs exploratory learning to enable a hardware robot to autonomously search for, navigate toward, and pick up a distant object as specified by a remote operator. In order to evaluate the viability of the learning, sensory integration, and decision-making models required for these tasks in both virtual and hardware versions of the Calliope robot, we created the CoCoRo (Cognitive Co-Robot) control system. Using CoCoRo, we demonstrate that through motor babbling of its wheels and arm, the Calliope is able to learn how to relate visual and proprioceptive information to achieve the hand-eye-body coordination required to complete its intended tasks.

The rest of this paper is arranged in the following way. Section Materials and Methods describes the CoCoRo architecture, the Calliope robotic platform used to evaluate the system, and a detailed description of the components used to achieve hand-eye-body coordination. Section Results presents the results of several experiments conducted to validate the reaching, navigation, and distant object retrieval goals. In Section Discussion, the methods and experimental results are discussed and compared to previous work. The paper concludes in Section Conclusion with a summary of the key contributions.

Materials and Methods

CoCoRo Architecture

CoCoRo uses a modular, synchronous architecture. It defines four types of system components: executive agent, sensorimotor devices, cognitive processes, and working memory. Each component in the system is chained together in serial with data flowing from one component to the next via a data structure termed a cognitive packet. A single iteration through all components is referred to as a cognitive cycle (**Figure 1**).

The cognitive cycle consists of four phases: executive, sensory, cognitive, and motor. In the executive phase, a cognitive packet is generated by the working memory component, which includes persistent information from the last cycle, time elapsed since the beginning of the previous cycle, and any commands from the executive agent. Next, in the sensory phase, all sensorimotor devices are polled to retrieve new raw sensory data. Then, in the cognitive phase, cognitive processes act on the sensory and memory data. Finally, in the motor phase, the sensorimotor devices execute any relevant motor commands generated from the previous phases. Finally, the executive agent is given the opportunity to store or transmit any data from the cognitive packet before the next one is generated and the cycle repeats.

The executive agent determines the broad goal objective and task the co-robot will perform. This could arise endogenously through a default behavior pattern or exogenously through commands received from a remote operator. The executive agent also has the ability to store or transmit data for later analysis or telepresence capabilities. Sensorimotor devices are elements that produce sensory data and/or execute motor commands, such as capturing image data from a camera or setting velocity commands to wheel motors. Cognitive processes are intended to be discrete, single purpose functions, such as detecting objects in

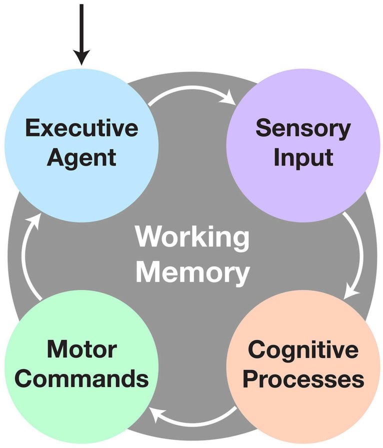


FIGURE 1 | The cognitive cycle. After initialization, the cognitive cycle runs until the user halts the robot. The executive agent checks for changes in goal directive, followed by acquisition of sensory data. Next comes processing of the data to fulfill the current objective. Finally any new motor commands are sent to the appropriate devices and the process repeats. All communication during and persistence across cycles is handled by the working memory system.

a visual scene or planning the motor actions needed to articulate a limb toward a desired target. These processes operate on either raw sensory data or the outputs of upstream processes. They then either output intermediate data for use by downstream processes or drive behavior in the form of motor commands. Finally, working memory retains persistent information over the duration of the designated task operation, such as what the goal target is, where it was last seen, and whether certain actions should be enabled or inhibited.

The CoCoRo architecture separates out the realization of a specific robotic platform from the cognitive control model by defining an API for writing the robot control system component modules and runtime programs. Using this approach, cognitive processes evaluated in a virtual environment can be directly applied to a real world robot without code changes—only the CoCoRo runtime, including operational parameters, and the sensorimotor device modules need be specific to a particular robot environment. Additionally, a common reference frame for working with various coordinate systems in three dimensions is also defined as part of this API to ensure consistent operation between components (**Figure 2**). All code was implemented using the Python programming language.

Robot Platform

The robot platform used in this study is the RoPro Calliope (**Figure 3**), a reference robot designed for the Tekkotsu robotics development environment (Tira-Thompson and Touretzky, 2011). The Calliope is a multimodal system consisting of an iRobot Create robot base mounted with a 7-degree-of-freedom (DOF) robotic limb and a Microsoft Kinect. All hardware components of the Calliope are centrally controlled via a laptop running Linux (Ubuntu 14.04) resting on top of the Create.

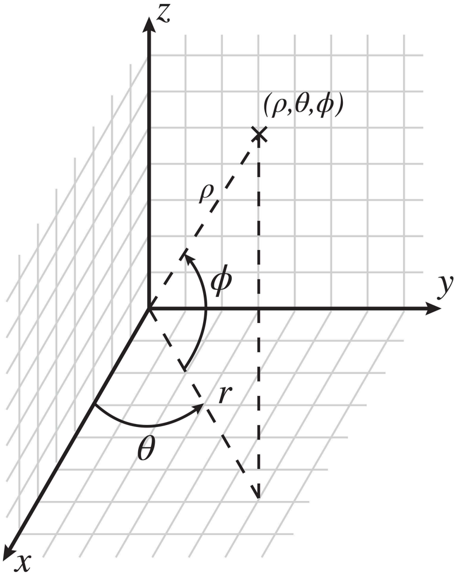


FIGURE 2 | The CoCoRo common coordinate reference frame. The origin is defined as the center of the robot's head. In Cartesian space, *x* is in front of the robot, with positive values going outward, *y* is the horizontal plane, with positive values going to the left, and *z* is the vertical plane, with positive values going up. In spherical space, ϱ is the distance from the origin to a given point, θ is the counterclockwise azimuth angle in radians, and ϕ is the inclination angle upward from the horizontal plane in radians.

The Create is a differential-drive robot with two drive wheels capable of up to 500 mm/s either forward or reverse and a third balancing wheel. The limb is constructed from Robotis Dynamixel servos and separated into a 4-DOF arm with horizontal shoulder, vertical shoulder, elbow, and wrist pitch joints on one servo network and a 3-DOF hand with wrist roll and two claws on another network. Each servo has 1024 addressable positions covering 300 degrees. The servos are controlled through a USB-to-TTL interface. The Kinect has a 640 × 480 32-bit color camera and a 640 × 480 12-bit depth camera. The cameras have a field of view of 1 radian horizontal and 0.75 radians vertical. The depth camera has an effective sensing range of 0.5–3.5 m. Pan and tilt control of the Kinect is provided by two additional Dynamixel servos also on the arm servo network. Power for the Kinect and arm servo network comes from a battery pack mounted on the back of the Create, while the hand servo network is powered from the Create's own battery. When fully assembled, the Calliope weighs 10.34 kg.

To enable safe testing and evaluation of the CoCoRo control system and component modules, a virtual representation of the Calliope was developed in Webots (Michel, 2004), a commercial mobile robot simulation software package. Webots allows for robot controllers to be written in a variety of languages including Python, which made it ideal for testing and evaluating the various CoCoRo components.

System Implementation

On top of the base CoCoRo platform we developed the components necessary to embody the Calliope with the ability to reach and grasp distant, visually identified objects. This task required the co-robot to perform the following coordination of subtasks: identify and localize objects in the environment, visually search for a desired object, navigate toward the object, reach for the object, and finally grasp the object in its hand.

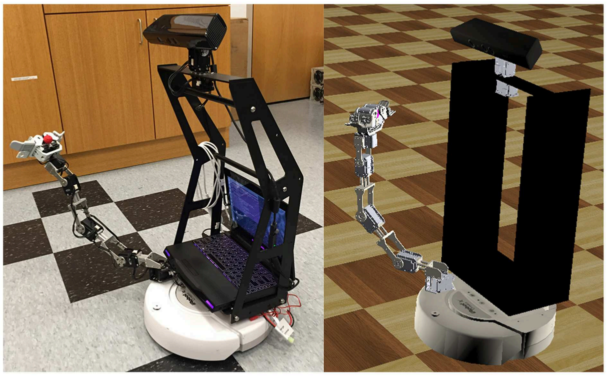


FIGURE 3 | The Calliope robot. The RoPro Calliope mobile robot (left) and its virtual counterpart in the Webots (http://www.cyberbotics.com/) robotics simulator (right).

The CoCoRo components created to fulfill this task include an executive agent that supported remote operator control; four sensorimotor device interfaces for the Calliope's Kinect, servos, and wheels; and multiple cognitive processes to perform decision-making and coordination for the various subtasks. The full cognitive cycle implementation is depicted in **Figure 4**.

The executive agent was implemented using the Asimov middleware system (Galbraith et al., 2011) to send and receive data between the Calliope and a remote operator. Operators were able to send goal directives and manual motor commands. They could also optionally receive video frames from the Calliope's camera. Additionally, the agent had the capability to store the contents of each cognitive packet to disk after the end of a cycle for later offline analysis.

Four sensorimotor devices were created: one for the Kinect, one for the Create, and one for each of the two servo networks. The Kinect device captured and provided the raw RGB and depth images while the Create device accepted and issued changes in wheel velocity. The servo devices, corresponding to the arm/neck and hand servo networks, provided the current positions of the joints, set the joint velocities and goal positions, and translated between CoCoRo's common reference frame and the internal Dynamixel reference frame.

The cognitive processes were divided into two functional groups: object awareness and motor planning. Object awareness consisted of two steps: detecting known objects in the visual scene and then localizing them in reference to the body. Motor planning contained the processes for generating and coordinating joint and wheel velocities to control head position, navigation, reaching, and grasping.

As employing robust computer vision methods to object detection was outside the scope of this work, we intentionally chose a simplistic approach. The robot used a color threshold method to detect predefined objects in a constrained environment. Objects were monochromatic cylinders and spheres defined by channel ranges in the CIELAB color space. CIELAB was chosen over RGB due to its greater robustness to changes in luminance. First the raw RGB image was converted to CIELAB using OpenCV and then segmented into a 5 × 5 grid of tiles. For each known object, the tile with the most matching pixels that fell into that object's color range was selected. The object was considered present if the pixel count exceeded a threshold of 64 pixels. The centroid of the object was then computed by taking the median x and y image coordinate values of all matching pixels. The depth value was selected by taking the corresponding pixel location from the depth image. Finally, these pixel values were added to the cognitive packet along with the object's label.

Object localization converted all detected objects from raw image coordinates (I_x, I_y, I_z) into relative egocentric locations (ρ, θ, Φ). The angular coordinates of each object were computed using the following transforms:

$$\theta = \left(\frac{1}{2} - \frac{I_x}{I_w}\right) F_h + \theta_p \tag{1}$$

$$\Phi = \left(\frac{1}{2} - \frac{I_y}{I_h}\right) F_v + \theta_t \tag{2}$$

Here, I_w and I_h were the image width and height in pixels, F_h and F_v were the horizontal and vertical fields-of-view, and θ_p and θ_t were the positions of the pan and tilt joints. This had the effect of converting raw pixel locations into retinotopic coordinates and then adjusting them based on the head position.

For the Kinect, I_z ranged from 0 to 2047, with 0 corresponding to >3.5 m, 2046 corresponding to approximately 0.5 m, and 2047 corresponding to an error code meaning no depth information was obtained. If an error code was detected, no value was set for ρ, otherwise it was computed by:

$$\rho = D(I_z) + l_n \sin(-\theta_t) \tag{3}$$

The first part transformed the Kinect pixel values to depths given in meters using function D adapted from Miller (2010). The second part adjusted for the tilt of the head away from center, where $l_n = 0.05$ m was the length of the neck.

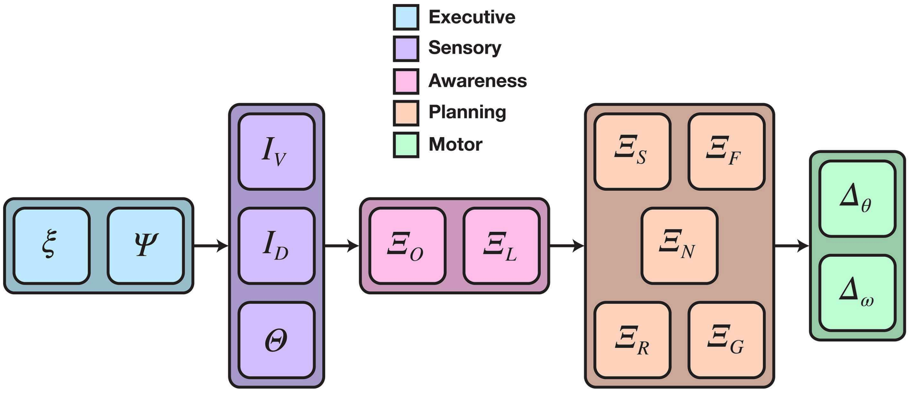


FIGURE 4 | Detailed cognitive cycle model for reaching and grasping distant objects. Data flows from left to right. Vertically aligned components could execute in parallel, though in practice all components execute in a single serial chain. The cognitive process phase was divided into two sub-phases: object awareness and motor planning. ξ is the cognitive packet, Ψ is the executive agent, I and Θ are data from camera and joint position sensors, respectively, Ξ is a cognitive process, and Δ is a motor command expressed as a joint or wheel velocity.

Once objects were detected and localized, they were passed on to the motor planning processes. Head position was determined by whether or not the goal object was detected in the visual scene. When the target was not detected, joint commands were generated to rotate the head in a fixed sweeping pattern to scan the environment until the target was found. Otherwise the robot fixated on the target by generating joint commands to position the head such that the target was held in the center of vision. For the scope of this work, no additional seeking behavior was implemented, so the robot remained stationary while scanning the environment indefinitely if the target could not be detected.

Reaching

Motor planning for reaching is based on the DIRECT model (Bullock et al., 1993; Guenther and Micci Barreca, 1997), which belongs to the class of psuedoinverse control methods for redundant manipulators (Klein and Huang, 1983). These methods solve the inverse kinematics problem of choosing appropriate joint velocities that achieve desired end-effector movement by computing the generalized psuedoinverse of the manipulator's Jacobian matrix.

There are two challenges to implementing this solution in practice. First is that the Jacobian matrix must be computable for all possible joint configurations. In stick models or simulations where the robot is treated as a rigid body and the exact geometry of the arm is known, the solution can be computed directly. For instance, the Calliope's limb (**Figure 5**) has the following ideal relationship between joint configuration and end effector's egocentric location:

$$\begin{aligned} x_e &= x_0 + \cos\theta_1(l_1 + l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3) \\ &\quad + l_4\cos(\theta_2+\theta_3+\theta_4)) \end{aligned} \tag{4}$$

$$\begin{aligned} y_e &= y_0 + \sin\theta_1(l_1 + l_2 cos\theta_2 + l_3\cos(\theta_2+\theta_3) \\ &\quad + l_4\cos(\theta_2+\theta_3+\theta_4)) \end{aligned} \tag{5}$$

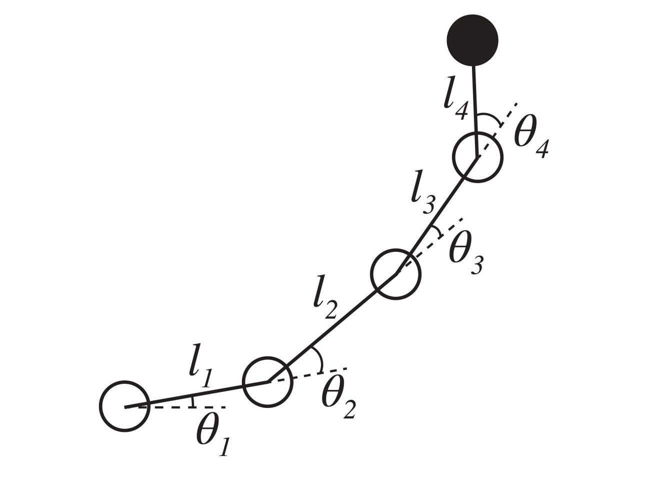


FIGURE 5 | Stick model of the Calliope arm. The Calliope arm has four revolute joints arranged in a linear chain. The first joint represents horizontal shoulder movement and rotates about the *z*-axis. The other three joints, vertical shoulder, elbow, and wrist pitch, respectively, rotate about the *y*-axis. The limb segment lengths are 0.11, 0.145, 0.138, and 0.135 m, respectively.

$$\begin{aligned} z_e &= z_0 + l_2\sin(\theta_2) + l_3\sin(\theta_2+\theta_3) \\ &\quad + l_4\sin(\theta_2+\theta_3+\theta_4) \end{aligned} \tag{6}$$

where (x_0, y_0, z_0) is the location of the base of the arm in the CoCoRo common reference frame, l_i is the length of the *ith* arm segment, and θ_i is the position of the *ith* joint. Using this, the Jacobian matrix and psuedoinverse can be easily derived and computed.

In the real world, however, the Calliope is susceptible to deviations from this model due to the invalidation of the rigid body assumption, operational limitations, and minor manufacturing defects. As such the error between the actual and computed Jacobian will vary in an inconsistent fashion across the workspace. This is further compounded by the second challenge to using the inverse kinematic model, which is determining where the hand is relative to the desired location.

Obtaining the value for the desired end-effector displacement, Δx, in a simulation could be as straightforward as tracking the allocentric coordinates of both end effector and desired target and then computing their difference. In an embodied system, where the robot can only act upon data from its sensors, arriving at an appropriate value for Δx is non-trivial. The desired reach target is located through the visual system, whereas the hand can be located through vision or, failing that, through an estimate achieved via proprioception. This latter modality is especially important, as the robot's hand may not be visible when reaching is initiated toward a target. Good hand-eye coordination, i.e., agreement between visual and proprioceptive position estimates, is important for obtaining consistent values of Δx and thus for maintaining smooth and effective reaching trajectories.

DIRECT addresses both the determination of the Jacobian and achieving good hand-eye coordination through neural network-based exploratory learning mechanisms. By motor babbling the joints in the arm and observing the resulting position of the end effector, the DIRECT neural network is able to learn the relationship between the visual and proprioceptive inputs. Using this method accounts for deviations from the idealized model by using actual data instead of theoretical predictions.

Our version of DIRECT is similar to that described in Guenther and Micci Barreca (1997) as we also use a hyperplane radial basis function (RBF) network (Stokbro et al., 1990; Du and Swamy, 2014) as our choice of neural network. However, we do not attempt to learn the inverse map, but instead only learn the forward map and then use it to numerically approximate the instantaneous Jacobian matrix. This is accomplished by querying the trained model for expected changes in end effector position due to slight perturbations of each joint in isolation. Once obtained, the arm joint motor plan is computed using the psuedoinverse method.

In addition to learning how to articulate its limb to reach for a particular location, the robot also needs to determine if that location is actually within its immediate reach, a task outside the scope of the DIRECT model. We have developed a solution to this reachability problem using the same motor babbling process employed by DIRECT. The reachability of a desired object is whether or not the robot can move its end effector to that exact location from its current position. Both the geometry of the robot's arm and the persistent features of its operational environment determine the reachable workspace of the robot, such as the robot's own body morphology and the relative position of the floor. An object is labeled as reachable if it is contained within a manifold encompassing all points that the end effector can move through. Defining this manifold is not achievable through simple polyhedral, however. Every place the hand can go is considered a reachable location; therefore all recorded locations of the hand are collected into a point cloud that represents a sampling of the reachability manifold. A Delaunay triangulation, a mesh of adjacent simplices, is then constructed from this set of points, which creates a convex approximation of the manifold. Additionally, like the RBF network, the Delaunay triangulation algorithm supports incremental update allowing it to be used in both offline and online learning scenarios. The test for reachability of an object becomes whether or not its location would fall within the boundaries of any simplex in the mesh. When a goal object is outside the range of reachability, the navigation system is disinhibited allowing wheel commands to be generated to move the robot toward the target as described in the next subsection. As soon as the object is deemed to be within reachable range, the navigation system is inhibited, preventing any further wheel movements.

Limited grasping capabilities were also implemented. For the purposes of this work, the actual grasping problem was reduced from 3DOF to 1DOF by making all grasping targets vertically aligned cylinders, e.g., soda cans. The wrist pose never had to change as it was always aligned for vertical targets, and the finger and thumb motor actions were treated as one synchronous motion to jointly open or close. The distance vector between the location of the hand and the target object that was computed during the reaching task was evaluated each cycle against a minimum grasping threshold. Once the hand was determined to be within this threshold for grasping the target, motor commands were issued to both close the hand at a fixed velocity and cease any new reaching-related joint velocities.

Motor Babbling

Motor babbling is an exploratory-based learning strategy for sensorimotor control. Through repeated execution of the action-perception cycle, an agent is able to build an internal model of how its motor behavior corresponds to sensory observations. The babbling aspect is that random actions are generated to explore and discover the range of possible outcomes with limited or no prior knowledge of what is actually possible. This strategy has been successfully used in neural network-based embodied learning for navigation (Zalama et al., 1995) and reaching (Bullock et al., 1993) using endogenously generated pseudorandom joint velocities. A drawback of those approaches, however, is that there is no active exploration of the workspace. Instead they passively rely on a large number of trials to fully cover the space. Recent approaches have explored an active form of motor babbling that either uses a confidence metric in accuracy to direct babbling to less confident regions (Saegusa et al., 2009) or a curiosity-driven reinforcement learning method that seeks out unexplored regions (Frank et al., 2014).

For this work, a semi-active approach was utilized. Endogenous random joint or wheel velocities were generated as in the passive case, but Sobol sequences (Sobol, 1976) were used instead of uniformly distributed pseudorandom numbers. A Sobol sequence is a set of quasi-random numbers designed to evenly cover a space for given sequence length. This provides a semi-active solution, as although it is still largely random, it is guaranteed that the babbling phase will result in actions that explore the entire workspace, thus reducing the number of training iterations required.

Navigation

The Calliope, owing to the iRobot Create base, uses a differential drive form of locomotion. Like with reaching, in order to navigate toward a desired target, the robot needs to solve the inverse kinematics problem of determining the wheel velocities that will move it to the appropriate location. Typically solved in allocentric, Cartesian space (Dudek and Jenkin, 2010), we present an egocentric, polar space solution that produces smooth trajectories.

Assuming constant wheel velocities ($\boldsymbol{v}_R$, $\boldsymbol{v}_L$) with no slippage over a fixed time interval, the inverse kinematic model is initially given as:

$$\begin{bmatrix} v_R \\ v_L \end{bmatrix} \Delta t = \begin{bmatrix} 1 & \frac{d_w}{2} \\ 1 & -\frac{d_w}{2} \end{bmatrix} \begin{bmatrix} s \\ \theta_R \end{bmatrix} \tag{7}$$

where d_w is the distance between the wheels and $\boldsymbol{s}$ is the desired trajectory arc length with angle of rotation θ_R. Determining (s, θ_R) is challenging when working in allocentric coordinates, where the robot must have a sense of the target location and its own relative to a fixed origin in the environment. This problem is avoided when working in egocentric coordinates, where the robot views everything in relationship to itself (**Figure 6**). The relationship between egocentric coordinates in the horizontal plane (r, θ) and the associated trajectory arc is:

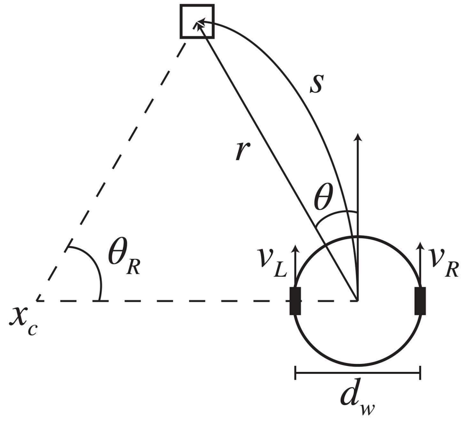


FIGURE 6 | Differential-drive kinematic model. Based on the visually determined relative location of the desired target (r, θ), the robot generated wheel velocities (v_L, v_R) to produce the trajectory arc that would reach the target. The arc has length s and angle of rotation θ_R about point x_c.

$$s = \frac{\theta r}{sin\theta} \tag{8}$$

$$\theta_R = 2\theta \tag{9}$$

By combing Equations (7–9) the egocentric inverse kinematics model is obtained:

$$v_R \Delta t = \left(\frac{\theta r}{sin\theta} + \theta d_w \right) \tag{10}$$

$$v_L \Delta t = \left(\frac{\theta r}{sin\theta} - \theta d_w \right) \tag{11}$$

In practice, however, the wheel velocities have maximum speeds (v_{Rmax}, v_{Lmax}) that this model does not accommodate; simply capping or scaling velocities that exceed these limits is insufficient as the difference between v_R and v_L is central to the desired trajectory movement and must be preserved. Let $\Delta t = 1$ s, $v_{Rmax} = v_{Lmax} = v_{max}$, and:

$$\delta = \frac{v_R - v_L}{2} = \theta d_w \tag{12}$$

then considering the imposed requirement of non-negative velocities, the wheel velocities are given by:

$$v_R = \max \left(\min \left(\frac{\theta r}{sin\theta}, v_{max} \right) - |\delta| + \delta,\ 0 \right) \tag{13}$$

$$v_L = \max \left(\min \left(\frac{\theta r}{sin\theta}, v_{max} \right) - |\delta| - \delta,\ 0 \right) \tag{14}$$

In egocentric space, the relative position of the target is continually changing while the robot is moving, so new velocities are generated every cycle. As no distinction needs to be made between stationary and moving targets as long as they can be localized, this method can produce smooth trajectories for both approaching a fixed location and pursuing a mobile object.

Results

The hand-eye-body coordination tasks were evaluated in three broad task areas: hand-eye coordination, egocentric navigation, and grasping distant objects (**Figure 7**). These experiments were conducted in both virtual and real world environments.

Hand-eye Coordination

The co-robot performed arm motor babbling to learn both the relationship between proprioceptive inputs of joint positions to the visual inputs of end-effector position and an approximation of the reachability manifold of the arm. Random target joint positions were generated over [−2.62, 2.62] radians per joint with velocities chosen to require 10 cognitive cycles to reach the new position. During this motor babbling phase, the co-robot fixated on its hand, identified by either a magenta circle (virtual) or red foam ball (real) attached to the end effector. If the end effector was visually located during a cognitive cycle, the arm joint positions and target location were recorded.

After the motor babbling phase ended, an offline training phase was conducted. Data outliers due to noise from the real world cobot were identified and rejected by detecting target positions with a nearest neighbor distance greater than 2.5 cm. A Delaunay triangulation was constructed from this data to approximate the reachability manifold.

A hyperplane RBF network was trained to learn the forward proprioceptive map. First a grid search was conducted using the collected data to determine the number of bases, Gaussian width, and learning rate to use for the network—the Gaussian centers were spread evenly across the joint input space of [−2.62, 2.62] radians per joint. Next, 10,000 distinct evenly spaced joint configurations and associate hand positions were generated from the rigid-body model of the arm (Equations 4–6) and used to prime the network. Finally, the network was trained on the collected data. To imitate online learning, data points were presented sequentially and only once.

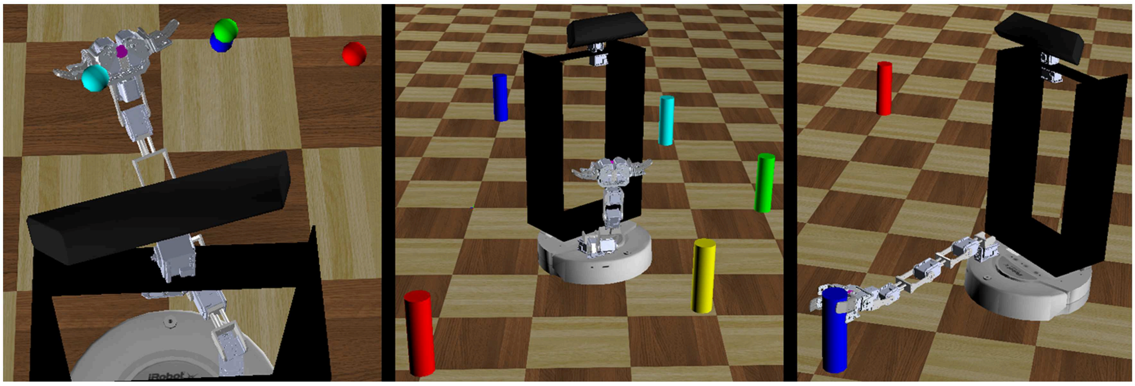


FIGURE 7 | Three robot behavioral experiments. The robot performed a series of behavioral tasks to evaluate the feasibility of the motor babbling approach. These tasks included repeatedly reaching to a series of targets in space (left), navigating toward a target and stopping within a set distance threshold (center), and grasping distant objects (right).

The network parameters chosen for both virtual and real world cobot were three bases per input dimension for a total of 3^4 or 81 bases, σ = 1.57, and α = 0.025. The network trained from within the virtual environment was able to reproduce the training set target positions with $R^2 = 0.926$ and RMSE = 0.051 while the network trained on the real world Calliope achieved $R^2 = 0.942$ and RMSE = 0.044.

The efficacy of the hand-eye coordination model acquired through motor babbling was then compared to that of one based on the rigid-body model. The virtual co-robot reached toward four colored targets suspended in the space in front of it in a predetermined order. The hand was deemed to have reached the target if the difference between detected positions was within (0.02, 0.034, 0.034) spherical units. Once reached, the co-robot moved to the next target in the sequence, completing the entire cycle three times. The position of the hand as determined by the robot was recorded and plotted (**Figure 8**).

Egocentric Navigation

Motor babbling of the wheels allowed the robot to learn the distance between its wheels. It fixated on a target initially placed 1.5 m directly in front of it, recorded the target's position provided from the visual system, then engaged each wheel at a fixed velocity selected from a Sobol sequence over [−0.15, 0.15] m/s for approximately 1 s. After the trial time had elapsed, the robot came to a halt, recorded the new relative position of the target, and computed the wheel distance estimate using:

$$\overline{d_w} = \frac{\Delta t}{\Delta\theta}(v_r - v_l) \tag{15}$$

It repeated this process using the reverse of the previously selected velocities to return to its approximate starting position. After several trials of forward and reverse pairs were conducted, the median of the estimates was taken as the robot's learned wheel distance (**Figure 9**).

Using the learned wheel distance, the robot navigated toward targets placed approximately 1 m away and at −90°, −45°, 0°, 45°, and 90° angles. The robot stopped once it determined it was within 20 cm of the target. Once the robot stopped moving, the actual distance between the edge of the target and the center of the robot was measured and recorded. The real and virtual robots achieved mean stopping distances of 22.7 ± 0.748 cm ($n = 15$) and 20.2 ± 0.458 cm ($n = 5$), respectively.

To demonstrate an example of human-robot interaction, the robot also followed a person identified by a held target object. The person started 1 m directly in front of the robot, holding the identifying object approximately 0.7 m off the ground. The person then walked in an 8 m perimeter square pattern just fast enough to prevent the robot from catching up. This was replicated in the virtual environment by having the target object hover above the ground and move on its own. During this task, the position of the target was smoothed using an exponential weighted moving average to mitigate sensor noise. Both the virtual and real world robots maintained pursuit over traversal of the pattern (**Figure 10**).

Grasping Distant Objects

The coordination of reaching and navigation was demonstrated in a task where the Calliope had to pick up an operator-directed target in the environment. The Calliope was placed in an environment with two (real) or three (virtual) known objects

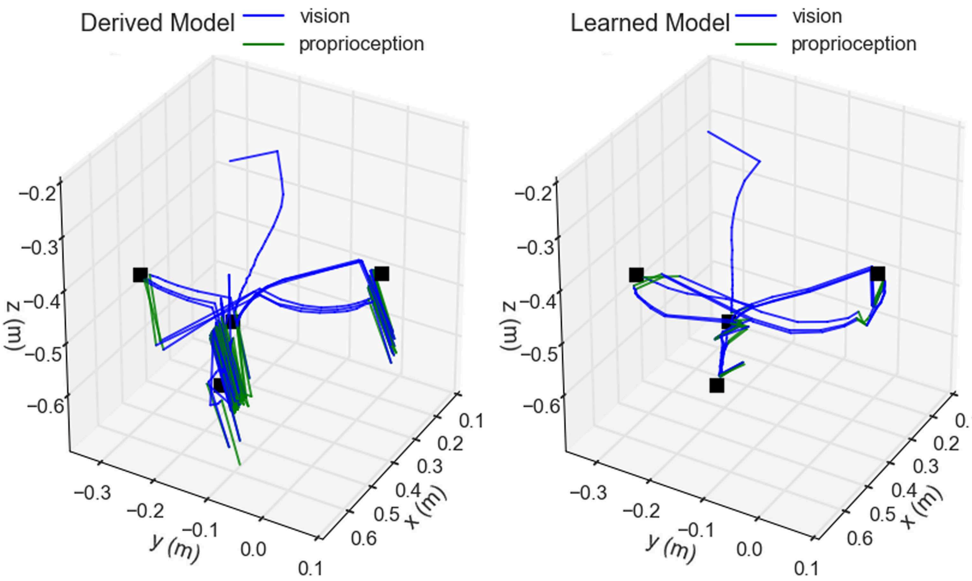


FIGURE 8 | Comparison of derived vs. learned models for hand-eye coordination. The trajectory of the hand positions as determined by the robot are shown during the execution of a reaching task cycling between four visually located targets (black). Blue components of the trace indicate when the hand was visually located, whereas green indicates when the proprioceptive model was used. Arm joint velocities were determined using Jacobian matrices either computed directly from the rigid-body model (left) or approximated from the trained neural network (right).

located at (1.5 m, 0°), (1.4 m, −45°), and (1 m, 45°) away, all outside the immediate grasping range of its arm. It was then activated and assigned one of the objects to find and pick up. The robot had to coordinate head position, wheel velocities, and arm and hand joint velocities to complete the task successfully (**Figure 11**). The virtual robot performed one trial for each target and managed to grasp and lift each for 100% completion. The real robot performed five trials for each target and successfully completed the task 80%, 40%, and 60% of the time, respectively, for an overall completion rate of 60%. In all cases where the Calliope failed to complete the task, it was because it grazed the target object with its hand, knocking it over. It still managed to stop within reaching distance and move its hand to the correct vicinity of the target. Videos of both virtual and real robots performing the task can be found in the Supplementary Materials.

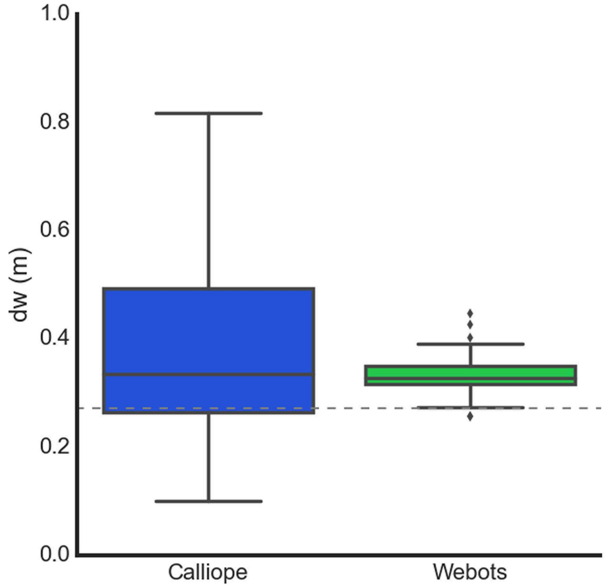


FIGURE 9 | Learning body size through motor babbling. The Calliope learned a $\overline{d}_W$ of 0.336 ± 0.088 m ($n = 91$), while the virtual robot learned a $\overline{d}_W$ of 0.326 ± 0.014 m ($n = 84$). The dotted line at 0.272 m represents the actual distance between the wheels.

Discussion

The CoCoRo Control System

One of the design choices with CoCoRo was to use a serial, synchronous data flow model. This was chosen for its relative simplicity of implementation and the ability to chain certain cognitive processes together in a defined order for coordination purposes. However, the penalty for using this architecture was that the entire cognitive cycle was rate-limited by the slowest component. This had no impact on the virtual environment where simulation time had no bearing on real time, but it did affect the real robot, where the object identification process proved slowest due to the naïve implementation of color matching applied to the relatively large input image. Many other robot platforms, including Tekkotsu and MoBeE (Frank et al., 2012), use threaded, finite state machine architectures, which can achieve real-time performance and take advantage

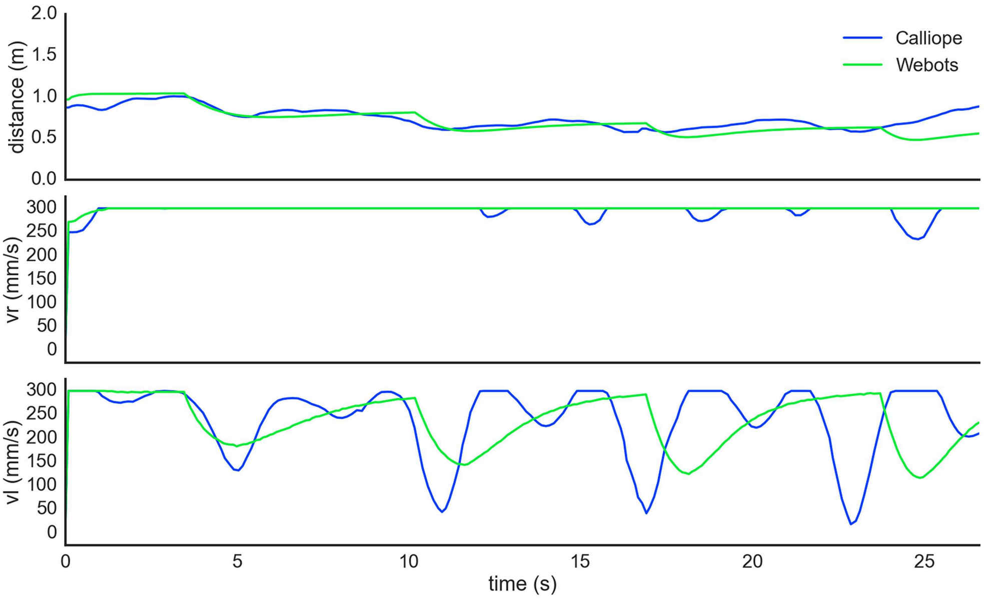


FIGURE 10 | Autonomous pursuit task. The robot visually tracked and pursued a target moving counterclockwise in a square pattern. The self-determined distance between the robot and target (top) slowly decreased as the robot got closer during the turns. Right wheel velocity (center) was kept at maximum while left wheel velocity (bottom) modulated during turns. The dips in both the right and left wheel velocities of the Calliope (blue) following a corner turn are from the robot overshooting and correcting itself.

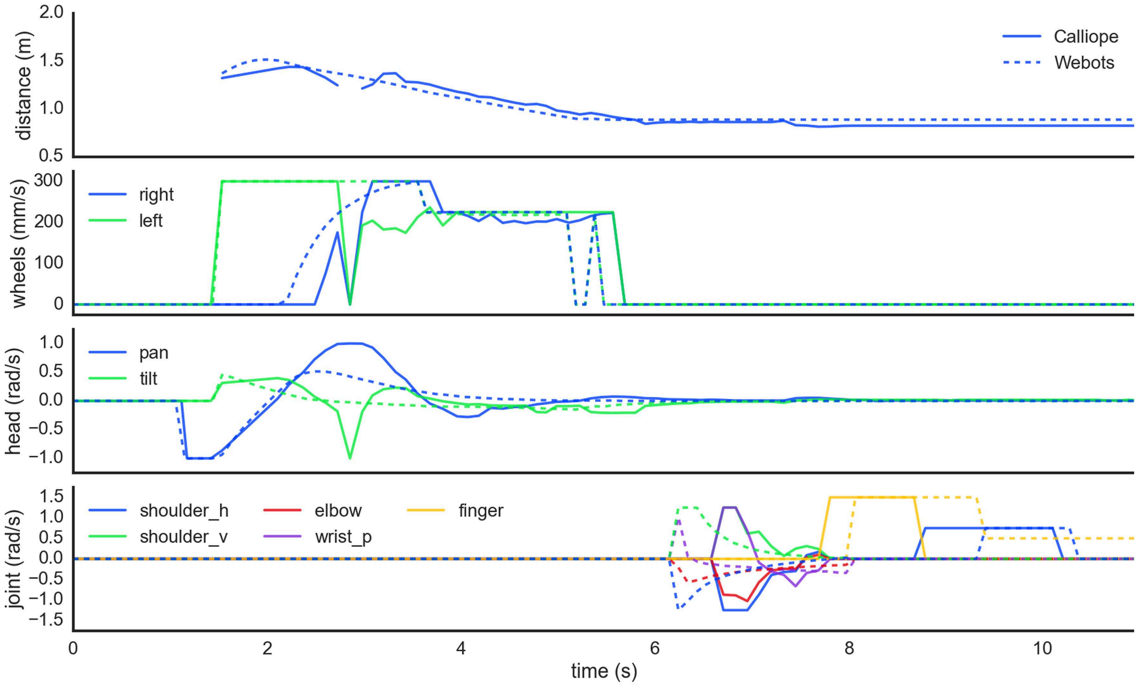
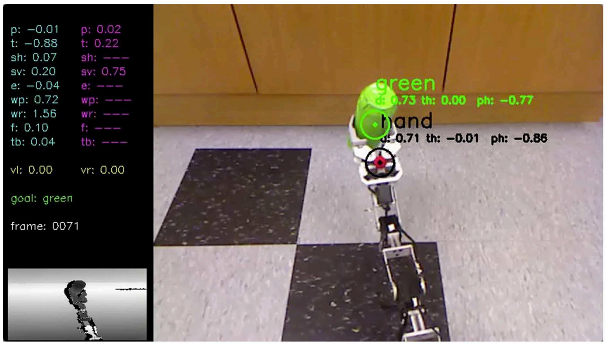


FIGURE 11 | Motor planning coordination while picking up a distant object. In order from the top, these plots show the detected distance to the target object followed by the generated wheel velocity, head position, and limb joint commands, respectively for both real (solid) and virtual (dashed) robots. First the robots scan the scene searching for the target. At 1.5 s, they locate the target to the right and navigate toward it while maintaining gaze fixation. Around the 5.5 s mark, the robots determine the object is reachable, stop navigation, and ensure head position is stable before starting to reach toward the target. Grasping is initiated around 8 s in and takes about 1.5 s to complete before the obtained target is finally lifted off the ground.

of concurrent and distributed processing of information. This avoids the rate-limiting problem of the serial architecture at the cost of increased system complexity. However, with the computational power inherent in modern laptops, like the one mounted on the Calliope, CoCoRo's simplistic structure did not interfere with the ability of the robot to complete tasks effectively. The Calliope operated at an average rate of 10 Hz during task execution, which was sufficiently fast enough to adjust motor commands as needed for the tasks undertaken albeit with the maximum wheel and joint velocities artificially reduced. Wheel velocities were capped at 300 mm/s and arm joint velocities were capped at ±1.5 rad/s. The simulation step time in Webots was set at the default value of 32 ms. As all sensor and motor component control steps must be a multiple of this simulation step, 96 ms was chosen to offer a comparable decision performance rate.

An additional benefit of using the serialized data flow model was the ability to easily capture and store the cognitive packet to disk, the data structure that contained all the sensory inputs, intermediate processing, and motor outputs from a given time point. This process was used extensively for both debugging purposes and offline analysis, such as providing the data for several of the figures in this paper. A tool was also created to reproduce robot point-of-view movies from these packets (**Figure 12**), which proved invaluable for tracking down issues with object detection and localization.


p: −0.01 p: 0.02
t: −0.88 t: 0.22
sh: 0.07 sh: ---
sv: 0.20 sv: 0.75
e: −0.04 e: ---
wp: 0.72 wp: ---
wr: 1.56 wr: ---
f: 0.10 f: ---
tb: 0.04 tb: ---

vl: 0.00 vr: 0.00

goal: green

frame: 0071

green
d: 0.73 th: 0.00 ph: −0.77
hand
d: 0.71 th: −0.01 ph: −0.86

FIGURE 12 | Calliope lifting an object. This is a frame taken from a movie (see Supplementary Materials) reconstructing the Calliope's point of view during a task to grasp and lift a green object initially located 1.5 m away. The movie is created from stored cognitive packets generated during the execution of the task and includes all sensory inputs, motor commands, and identified objects.

Virtual Environments

The use of simulations and virtual environments are key to developing and evaluating robotic control systems. If the virtual environment provides a good enough approximation of the real environment, certain tasks can be bootstrapped in the simulation first, such as building up the internal neural network weights for control tasks. These weights can then be transferred directly to the physical co-robot which would then need a shorter recalibration learning session than if it had started with untrained networks. We used just such a method in training the neural network responsible for reaching. Training it first using an idealized set of inputs to outputs primed the network and provided reasonable results for locations in the reaching space that were not obtainable through motor babbling alone, i.e., where vision failed to detect the hand. The later data collected from motor babbling was then able to retrain the network to be more in line with the actual observed results instead of those generated by the rigid-body approximation.

However, we also encountered several discrepancies when moving between virtual and real sensors. The images from the virtual Kinect were always crisply rendered, whereas the images being pulled from the real Kinect were susceptible to noise. The sources of noise included motion blur introduced by movement from the body and head, and potential changes in luminance due to automatic white balancing performed by the Kinect video camera. The depth camera in the virtual environment, like the virtual video camera, was generated from the OpenGL buffer directly and did not suffer the effect of infrared shadows. These shadows were areas visible in the video image but in which no depth information could be obtained due to objects in the foreground preventing the infrared signals from reaching them. Despite these challenges in using video and depth image data in the real environment, the Calliope was still able to perform at a high level for the tasks explored, though additional checks had to be added for cases in which objects were visible but no depth information could be obtained.

Likewise, the behavior of servos varied between simulation and reality. In the virtual environment, servos would move smoothly in response to any requested velocity within defined operational range and supported high precision positional accuracy. The real servos, on the other hand, were limited by having only 1024 addressable positions for a resolution of about 0.005 radians. This contributed to occasional jittery behavior when attempting to hold joints in a particular pose due to the effects of rounding. The real servos also did not support specifying a velocity of zero to halt movement. Instead, we had to rely on a combination of velocity and positional control to achieve a fixed joint configuration. Finally, the skeleton of the arm itself contained screws prone to loosening during continual operation, resulting in slight changes to the position of the end effector over time.

Hand-eye Coordination

Controlling redundant joint manipulators is an open challenge in robotics, as closed-form analytic solutions to the inverse kinematics problem may not exist. Feedback-based control strategies have proven successful, but require reasonably accurate sensors to provide the needed error signals. These can be difficult to acquire for a non-planar limb outside of simulation or highly controlled workspaces. As a requirement of co-robots is to operate in largely uncontrolled environments, the control system should not rely on external sensors and fixed workspaces. We used a variant of the DIRECT model, a biologically inspired neural network approach to feedback-based control of a limb. Desirable features of DIRECT that make it useful for co-robots are that it is egocentric, so all sensor information comes from its own perspective, and it can adapt to changes in limb configuration. However, DIRECT, like many other solutions, was validated in simulation using perfect knowledge of end-effector position and stick-model limbs. Other applications of DIRECT have been reported (Vilaplana and Coronado, 2006; Grosse-Wentrup and Contreras-Vidal, 2007; Bouganis and Shanahan, 2010), but these too were only performed in simulation with perfect positional knowledge and lack of physical constraints beyond joint rotation boundaries. Our implementation is the first instance we are aware of that demonstrates the efficacy of DIRECT using actual computer vision to determine end-effector and target localization. Furthermore, this is also the first demonstration of DIRECT embodied in a real-world robot working in a 3D workspace.

Using visual inputs from a camera and working with a physical robot presented its own set of challenges for DIRECT, computer vision not with standing. DIRECT uses motor babbling to learn the space of movements, so it must be able to observe the end-effector in order to learn how it moves in a particular part of the workspace. With a fixed camera vantage point, body components obstructing views, and limitations of the camera sensor, the Calliope had several blind spots. Our solution to this was to prime the network before motor babbling commenced using the rigid-body model of the arm to generate thousands of training points evenly spaced across the entire hypothetical workspace. The network was trained using a learning rate an order of magnitude lower than that used during motor babbling so that real observed data would take precedence.

For instance, the observed location of the robot's hand in the virtual environment displayed close similarity to that of the rigid-body model for the portion of the workspace the arm was able to reach during motor babbling (**Figure 13**). As can be seen, this actually represented only a fraction of the theoretical range if the arm was free of any obstacles. The use of an identifying color marker on the top of the hand also produced a compressed range of visible locations. If the configuration on the arm resulted in the hand positioned upside down, for instance, it would not be recognized.

The major difference between theoretical and detected position came in the real world Calliope, where the detected distance of the hand was almost 10 cm on average closer than the model would predict. This can be attributed to two factors: a greater offset from the location of the visual marker to the end of the hand and the less precise distance estimation from the actual Kinect's depth camera vs. the simulated Kinect. The observed range of motion for the real hand was even more compressed than the virtual one, however, due to the Kinect's blindness within close proximity. Relying solely on either observed data or theoretical model would have produced large gaps or erroneous estimates, respectively. Using the theoretical model for initially priming the RBF neural network then further training with the motor babbling results provided a solution that enabled the use of both approaches to complement each other.

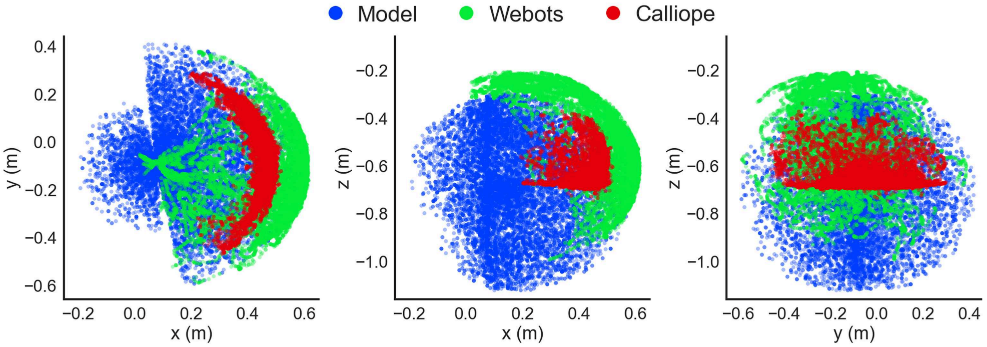


FIGURE 13 | Detected hand position during motor babbling. Recorded hand positions are shown in the xy- (left), xz- (center), and yz-planes (right). A kinematic stick model (blue) computed hand positions using randomly generated joint positions and the geometry of the arm, while both the Webots virtual environment simulation (green) and Calliope (red) used visual information to determine hand position during a motor babbling task.

For actually generating joint trajectories during a reach task, the analytically determined Jacobian from the rigid-body model produced similar behavior to that approximated by the trained neural network in three of the four reaching segments. The rigid-body model, based in Cartesian coordinates, produced straighter trajectories between targets but had significant disagreement between its visual and proprioceptive locations as exhibited by the trajectory shifts when switching between the modalities occurred. This most impacted the model during the downward trajectory from target three to four, where it got stuck and convulsed for several seconds before finally achieving a correct configuration. This was due to the first target, placed just above and in front of the fourth, occluding the marker on the hand toward the end of the trajectory resulting in the model flipping between visual and proprioceptive locations. The disagreement between the two was large enough that, when using visual input, the hand was perceived where it actually was, above the target, but when using proprioception, the hand was perceived to be below the target. This conflict produced the observed spasms. While all targets were eventually reached here, in a separate instance the arm became locked into a never-ending cycle of jittering up and down and the trial had to be terminated. The neural network model, by contrast, was based in spherical coordinates, produced slightly arced trajectories, and had much greater agreement between proprioception and vision. It experienced no difficulties in any of the reaching segments. Even when losing sight of the hand, there was enough agreement in the two modalities to allow for consistent smooth behavior during the trials.

Egocentric Navigation

In determining wheel distance, the real and virtual robots produced very similar final estimates, with the main difference being the noisiness of the Calliope's samples. Both robots were over the actual distance by 6 and 5 cm, respectively. This error could be related to the relative distances between wheels, camera, and reference target, as extending the wheel distance out further in the virtual environment produced very accurate estimates. This error did not appear to have an impact on the actual navigation tasks, as both the stationary and pursuit tasks produced comparable results. In the stationary task, the difference in average stopping distance was only 2.5 cm, while in the pursuit task, the Calliope performed well despite slightly overshooting the turns then having to correct.

This method for egocentric navigation employs an aiming strategy (Franz and Mallot, 2000) for local navigation, where the goal of path planning is to keep a desired target position directly in front of the robot while moving toward it. Other aiming approaches include Concentric Spatial Maps (CSM) (Chao and Dyer, 1999), which uses a neural network to store goal positions and obstacles in discrete locations arranged in concentric circles around the agent. A similar, though non-neural, approach to CSM is used to produce multi-agent pedestrian navigation through crowds (Kapadia et al., 2012). Both of these methods account for obstacles whereas we assumed a clear path. CSM, however, requires the environment map be loaded *a priori*, while the pedestrian model does not use sensory information from the agents themselves and instead determines them from the global simulation state.

An alternative and complementary strategy to aiming is guidance (Franz and Mallot, 2000), where the relative positions of environmental cues are used to determine desired trajectories. Examples of guidance-based approaches include ENav and variants (Altun and Koku, 2005; Fleming, 2005). They are based on the sensory egosphere (SES) (Albus, 1991), a 2D spherical projection of incoming sensory data to a spatial representation of the agent's environment, where the goal is to match the angular displacements of visually identified landmarks in the current SES with those provided in the desired SES. ENav is the only other method we are aware of to have reported implementation attempts outside of simulation (Fleming, 2005), though with limited results.

These navigation methods provide path planning abstracted from a specific kinematic model of locomotion. While ostensibly more general, they may produce trajectories that are not possible by an actual mobile robot, so an appreciation for the inverse kinematics of locomotion for target robot platforms is critical to produce a model that can work in real environments.

For differential-drive navigation, the inverse kinematics problem can be solved by breaking down the desired trajectories into pairs of distinct motions: first rotate in place to face the target, then drive straight forward toward it (Dudek and Jenkin, 2010). This, however, produces jerky motion, requiring the robot to stop forward progress every time it needs to rotate. For a clear path in an ideal environment, the expectation would be only one rotation and one direct forward trajectory. However, in a real world environment, wheel slippage, dynamic target location, and perturbations in the floor can result in deviations from the ideal trajectory, requiring compensatory corrections, each resulting in the robot having to stop, rotate, and begin forward again. This would be especially inefficient in the egocentric model, where the relative positions of objects are always changing as the robot moves.

Similar arc-based solutions to the one above have been proposed in both Cartesian (Bethencourt et al., 2011) and polar (Maulana et al., 2014) forms, though the former relies upon accurate accumulation of encoder data to reconstruct allocentric position while the latter is geared toward following a fixed track. Instead of learning just the body size as demonstrated here, the NETMORC model (Zalama et al., 1995) attempts to learn the inverse kinematic solution itself through a neural network trained via a similar motor babbling phase. However, only simulated results with perfect positional information used in training the network were reported.

This is the first work we are aware of that combines the use of egocentric navigation with a specific model of inverse kinematics. Not only does this approach succeed with a high accuracy in simulation, it works very well in a real world robot despite the increased noise from and limitations of actual hardware and environments.

Grasping Distant Objects

The task of grasping and lifting distant objects combines the previously described subtasks into a unified whole, requiring an additional layer of coordination on top of the individual motor plans. The motor planning coordination strategy used in this work was to take a largely lock-step approach, where the individual subtasks were disinhibited only when their role was called upon. The only exception to this was head movement, which operated in parallel to the progression of navigation, reaching, grasping, and lifting. This coordination was implemented by having each cognitive process in the chain alter or check the working memory system and inhibiting or disinhibiting itself based on its state.

Two main factors can be attributed to the cases where the real robot failed to complete the grasping task by knocking the target over. First is the simplistic object identification method, which is highly susceptible to noise and treats objects as points. This results in generally poor performance when precision adjustments were needed, which were typically required due to the second factor, the segregated process of only reaching once navigation stopped. In this arrangement, the arm is held out and to the side until the reaching subtask begins. It makes a downward arcing trajectory to reach the target, which can result in the hand clipping the side of the object if the robot is even a centimeter too close. If the hand began its reach earlier while the robot was still driving forward, the hand could be brought into position before there was a risk of inadvertent contact.

Other approaches to visually guided mobile manipulators employ more fluid motor control and coordination (Andaluz et al., 2012a,b; Kazemi et al., 2012). Related to the co-robot goal of working in unstructured environments, (Xie et al., 2014) presents a model for visual-guided control for grasping household items. All of these systems use a camera mounted on the end-effector instead of elsewhere on the body. These eye-in-hand visual servoing systems can achieve greater grasping and manipulation accuracy at the expense of having to manage a potentially highly articulated neck, i.e., the arm itself, when not engaged in an actual reach action. They also lack the flexibility of the alternate hand-to-eye approach used by the Calliope.

The simplistic method for visual object detection worked well enough for both reaching and navigation in the virtual environment where color detection is much easier. It was less effective in the real world as it was highly susceptible to noise. For navigation, which operated in 2D, this proved less of an issue, but it did impact the success of reaching and grasping, which required accurate 3D locations. The grasping method used was also the simplest available. Real world use would require more intelligent grasping algorithms for shaping the hand to accommodate a variety of object shapes. As CoCoRo supports drop in replacement of components, upgrading to more robust computer vision and grasping processes would be possible.

The egocentric model worked well for traversing the immediate vicinity of the robot assuming a clear path to the target destination. If any obstacles were in its path that did not occlude the target object, however, the robot would attempt to drive through them. Likewise, if the robot failed to detect the desired target in its sensory field, it would either have to revert to an allocentric representation to derive new egocentric coordinates from memory or engage in some form of directed search.

Conclusion

We presented a control system with an eye toward co-robots that used motor babbling to enable a robot to learn about aspects of its own configuration in regards to hand-eye-body coordination. This system was built on a software platform designed to enable modular evaluation of the learning, sensory processing, and decision-making motor components across both virtual and physical versions of the Calliope robot. The capabilities embodied in the robot enabled it to autonomously follow a person around a room and retrieve distant objects specified by a remote operator. In order to achieve this we demonstrated a variant of the DIRECT neural model for reaching in a hardware robot and complemented it with novel methods for determining if the intended reach target is actually within the robot's grasp and a means for egocentric-based navigation to drive it toward the target if it isn't.

There is still significant work to be done in order to extend this initial system to more practical real-world co-robot use. Adapting to cluttered and dynamic environments would require a much more robust and powerful form of visual object detection and identification that the simplistic model currently used. The navigational system would also be extended to handle obstacle avoidance and combine allocentric and egocentric path planning strategies. Smooth concurrent motor control coordination would also be a desirable improvement over the current lock-step approach.

Acknowledgments

This work was supported by the Center of Excellence for Learning in Education, Science, and Technology, a National Science Foundation Science of Learning Center (NSF SMA-0835976).

Supplementary Material

The Supplementary Material for this article can be found online at: http://journal.frontiersin.org/article/10.3389/fnbot.2015.00007

References

Albus, J. S. (1991). Outline for a theory of intelligence. *IEEE Trans. Syst. Man, Cybern.* 21, 473–509. doi: 10.1109/21.97471

Altun, K., and Koku, A. B. (2005). "Evaluation of egocentric navigation methods," in *IEEE International Workshop on Robot and Human Interactive Communication, 2005 (ROMAN 2005)*, 396–401.

Andaluz, V., Carelli, R., Salinas, L., Toibero, J. M., and Roberti, F. (2012a). Visual control with adaptive dynamical compensation for 3D target tracking by mobile manipulators. *Mechatronics* 22, 491–502. doi: 10.1016/j.mechatronics.2011.09.013

Andaluz, V., Roberti, F., Toibero, J. M., and Carelli, R. (2012b). Adaptive unified motion control of mobile manipulators. *Control Eng. Pract.* 20, 1337–1352. doi: 10.1016/j.conengprac.2012.07.008

Bethencourt, J. V. M., Ling, Q., and Fernandez, A. V. (2011). "Controller design and implementation for a differential drive wheeled mobile robot," in *Control and Decision Conference (CCDC), 2011 Chinese* (Mianyang: IEEE) 4038–4043. doi: 10.1109/CCDC.2011.5968930

Bouganis, A., and Shanahan, M. (2010). "Training a spiking neural network to control a 4-DoF robotic arm based on spike timing-dependent plasticity," in *Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN)* (Barcelona: IEEE), 1–8. doi: 10.1109/IJCNN.2010.5596525

Bullock, D., Grossberg, S., and Guenther, F. H. (1993). A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm. *J. Cogn. Neurosci.* 5, 408–435. doi: 10.1162/jocn.1993.5.4.408

Chao, G., and Dyer, M. G. (1999). "Concentric spatial maps for neural network based navigation," in *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN 99)* (Edinburgh), 144–149.

Du, K.-L., and Swamy, M. N. S. (2014). "Radial basis function networks," in *Neural Networks and Statistical Learning* (London: Springer), 299–335.

Dudek, G., and Jenkin, M. (2010). *Computational Principles of Mobile Robotics, 2nd Edn.* New York, NY: Cambridge University Press.

Fleming, P. (2005). *Implementing a Robust 3-Dimensional Egocentric Navigation System.* MS thesis, Vanderbilt University.

Frank, M., Leitner, J., Stollenga, M., Förster, A., and Schmidhuber, J. (2014). Curiosity driven reinforcement learning for motion planning on humanoids. *Front. Neurorobot.* 7:25. doi: 10.3389/fnbot.2013.00025

Frank, M., Leitner, J., Stollenga, M., Harding, S., Foerster, A., and Schmidhuber, J. (2012). "The modular behavioral environment for humanoids and other robots (mobee)," *in 9th International Conference on Informatics in Control, Automation and Robotics (ICINCO)* (Rome).

Franz, M. O., and Mallot, H. A. (2000). Biomimetic robot navigation. *Robot. Auton. Syst.* 30, 133–153. doi: 10.1016/S0921-8890(99)00069-X

Galbraith, B., Chandler, B., and Versace, M. (2011). "Asimov: middleware for modeling the brain on the irobot create," in *PyCon* 2011 (Atlanta, GA).

Grosse-Wentrup, M., and Contreras-Vidal, J. L. (2007). The role of the striatum in adaptation learning: a computational model. *Biol. Cyb.* 96, 377–388. doi: 10.1007/s00422-007-0142-8

Guenther, F. H., and Micci Barreca, D. (1997). "Neural models for flexible control of redundant systems," in *Self-organization, Computational Maps, and Motor Control*, eds P. G. Morasso and V. Sanguineti (Amsterdam: North Holland), 383–421.

Hayes, B., and Scassellati, B. (2013). "Challenges in shared-environment human-robot collaboration," in *Proceedings of the Collaborative Manipulation Workshop at the ACM/IEEE International Conference on Human-Robot Interaction (HRI 2013)* (Tokyo).

Kapadia, M., Singh, S., Hewlett, W., Reinman, G., and Faloutsos, P. (2012). Parallelized egocentric fields for autonomous navigation. *Visual Comp.* 28, 1209–1227. doi: 10.1007/s00371-011-0669-5

Kazemi, M., Gupta, K., and Mehrandezh, M. (2012). "Path planning for image-based control of wheeled mobile manipulators," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Vilamoura), 5306–5312.

Klein, C. A., and Huang, C.-H. (1983). Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Trans. Syst. Man, Cybern.* 13, 245–250. doi: 10.1109/TSMC.1983.6313123

Kristoffersson, A., Coradeschi, S., and Loutfi, A. (2013). A review of mobile robotic telepresence. *Adv. Hum. Comp. Interact.* 2013, 1–17. doi: 10.1155/2013/902316

Maulana, E., Muslim, M. A., and Zainuri, A. (2014). "Inverse kinematics of a two-wheeled differential drive an autonomous mobile robot," in *2014 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, 93–98.

Michel, O. (2004). WebotsTM: professional mobile robot simulation. *Int. J. Adv. Rob. Syst.* 1, 39–42. doi: 10.5772/5618

Miller, A. (2010). "calibkinect.py." Accessed April 22, 2013. Available online at: https://github.com/amiller/libfreenect-goodies/blob/master/calibkinect.py

Saegusa, R., Metta, G., Sandini, G., and Sakka, S. (2009). "Active motor babbling for sensorimotor learning," in *IEEE International Conference on Robotics and Biomimetics, 2008 (ROBIO 2008)* (Bangkok: IEEE), 794–799. doi: 10.1109/ROBIO.2009.4913101

Sobol, I. M. (1976). Uniformly distributed sequences with an additional uniform property. *USSR Comput. Math. Mathemat. Phys.* 16, 236–242. doi: 10.1016/0041-5553(76)90154-3

Stokbro, K., Umberger, D. K., and Hertz, J. A. (1990). Exploiting neurons with localized receptive fields to learn chaos. *Compl. Syst.* 4, 603–622.

Tira-Thompson, E., and Touretzky, D. S. (2011). "The Tekkotsu robotics development environment," in *IEEE International Conference on Robotics and Automation (ICRA)* (Shanghai: IEEE), 6084–6089. doi: 10.1109/ICRA.2011.5980533

Vilaplana, J. M., and Coronado, J. L. (2006). A neural network model for coordination of hand gesture during reach to grasp. *Neural Netw.* 19, 12–30. doi: 10.1016/j.neunet.2005.07.014

Xie, H., Li, G., Wang, Y., Fu, Z., and Zhou, F. (2014). Research on visual servo grasping of household objects for nonholonomic mobile manipulator. *J. Control Sci. Eng.* 2014, 1–13. doi: 10.1155/2014/315396

Zalama, E., Gaudiano, P., and Coronado, J. L. (1995). A real-time, unsupervised neural network for the low-level control of a mobile robot in a nonstationary environment. *Neural Netw.* 8, 103–123. doi: 10.1016/0893-6080(94)00063-R

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: 17 August 2015
doi: 10.3389/fnbot.2015.00009

Reward-modulated Hebbian plasticity as leverage for partially embodied control in compliant robotics

Jeroen Burms[1], *Ken Caluwaerts*[1,2] *and Joni Dambre*[1]*

[1] Computing Systems Laboratory (Reservoir Team), Electronics and Information Systems Department (ELIS), Ghent University, Ghent, Belgium, [2] Intelligent Robotics Group, NASA Ames Research Center, Oak Ridge Associated Universities, Moffett Field, CA, USA

In embodied computation (or morphological computation), part of the complexity of motor control is offloaded to the body dynamics. We demonstrate that a simple Hebbian-like learning rule can be used to train systems with (partial) embodiment, and can be extended outside of the scope of traditional neural networks. To this end, we apply the learning rule to optimize the connection weights of recurrent neural networks with different topologies and for various tasks. We then apply this learning rule to a simulated compliant tensegrity robot by optimizing static feedback controllers that directly exploit the dynamics of the robot body. This leads to partially embodied controllers, i.e., hybrid controllers that naturally integrate the computations that are performed by the robot body into a neural network architecture. Our results demonstrate the universal applicability of reward-modulated Hebbian learning. Furthermore, they demonstrate the robustness of systems trained with the learning rule. This study strengthens our belief that compliant robots should or can be seen as computational units, instead of dumb hardware that needs a complex controller. This link between compliant robotics and neural networks is also the main reason for our search for simple universal learning rules for both neural networks and robotics.

Keywords: compliant robotics, Hebbian plasticity, morphological computation, recurrent neural networks, tensegrity

Edited by:
Andrea Soltoggio,
Loughborough University, UK

Reviewed by:
Jean-Baptiste Mouret,
INRIA, France
Joshua E. Auerbach,
École Polytechnique Fédérale de Lausanne, Switzerland
Hidenobu Sumioka,
Advanced Telecommunications Research Institute International, Japan

Correspondence:
Joni Dambre,
Sint-Pietersnieuwstraat 41,
Ghent, Belgium
joni.dambre@ugent.be

Received: *27 January 2015*
Accepted: *29 July 2015*
Published: *17 August 2015*

Citation:
Burms J, Caluwaerts K and Dambre J (2015) Reward-modulated Hebbian plasticity as leverage for partially embodied control in compliant robotics.
Front. Neurorobot. 9:9.
doi: 10.3389/fnbot.2015.00009

1. Introduction

Hebbian theory has been around for over half a century (Hebb, 1949), but it still sparks the interest of today's researchers. Small changes to the basic correlation learning rule result in various well-known algorithms, such as principal (Oja, 1982; Sanger, 1989) or independent component (Hyvrinen and Oja, 2000; Clopath et al., 2008) extractor networks. The basic rule is biologically plausible as are some of its variations (Mazzoni et al., 1991; Loewenstein and Seung, 2006). Whereas all these approaches belong to the general category of unsupervised learning, *reward modulated Hebbian* (RMH) learning is similar to reinforcement learning in that it can be used to tune a neural system to solve a specific task without the need to know the desired output signals at the neural level (Fiete and Seung, 2006; Legenstein et al., 2010; Hoerzer et al., 2012; Soltoggio and Steil, 2013; Soltoggio et al., 2013). When using RMH learning in a robotics context, a reward can be computed, e.g., by comparing the sensory inputs with the desired observations. The use of RMH learning for optimizing robot motor control has several additional advantages. First, the basic learning rule is simple. There is no need for complex mathematical operations and it can therefore be efficiently implemented on various platforms in hardware and software. Second, it allows for a distributed implementation: a central unit can be responsible for a global reward, which can then be broadcast to the learning units of local controllers. Finally, RMH learning is an online learning approach. If the reward mechanism remains active, the controller can adapt to changes in the robot morphology or dynamics, e.g., due to wear or damage.

Class one tensegrity structures (Skelton and de Oliveira, 2009; Caluwaerts et al., 2014) consist of compression members held together by tension members in such a way that compression members are never directly connected. In robotics, these are typically a set of rods, interconnected by tension elements (springs or cables) between the rods' endpoints. These structures can serve as compliant robot bodies, by allowing some or all of the tension elements to be actuated. This results in flexible pin-jointed structures that make efficient use of materials, and are both extremely robust and lightweight (Caluwaerts et al., 2014). Tensegrities have also been researched from various other perspectives, from architecture and art (Snelson, 1965) to mathematics (Connelly and Back, 1998) and even biology (Ingber, 1997).

In previous work (Caluwaerts et al., 2012), we demonstrated that the motor control of a tensegrity robot can be drastically simplified by using its body as a computational resource. This approach originated from the concepts of *physical reservoir computing* (Verstraeten et al., 2007) and *morphological computation* (Pfeifer and Bongard, 2007), both of which treat the use of physical systems or *bodies* as a computational resource in so-called embodied computation. In Caluwaerts et al. (2012), we mainly focused on approximating motor signals through a single layer linear neural network acting as a feedback controller. The flexibility and lack of joints of our tensegrity robot allowed for simple learning rules, as the risk of failure due to mechanical stress or hard constraints was minimal. As a consequence, the feedback weights were learned by applying online supervised learning rules to approximate the target motor signals, among which was a supervised version of RMH learning.

Various forms of RMH learning rules have already been extensively studied in the context of both spiking (Fiete and Seung, 2006; Izhikevich, 2007; Legenstein et al., 2008) and rate-based (Loewenstein and Seung, 2006; Loewenstein, 2008; Soltoggio and Steil, 2013; Soltoggio et al., 2013) neural networks. The learning rule we handle uses noise as an exploratory term, similar to Legenstein et al. (2010), and can be shown to approximate gradient descend (Fiete and Seung, 2006). In this paper, we show that the RMH learning rule can be extended to systems exhibiting partial embodiment, i.e., agents that actively see and use their body as a computational resource. In these partially embodied systems, the computations that are performed by the body are naturally integrated into the controller architecture. We use the term "partially" to make the distinction with full embodiment, where agents do not need a controller, and with "trivial" embodiment, where little to no computations are offloaded to the body.

We first consider various analog recurrent neural network tasks and setups. Second, we will demonstrate that the RMH learning rule can be carried over beyond the scope of neural networks. We train the linear feedback weights of a secondary controller in a two-level control hierarchy for end-effector control in a highly compliant, simulated class one tensegrity robot. The primary controller – a simple feed forward kinematic controller – generates control signals derived from a very rough static inverse model of the relationship between end-effector positions and actuator signals. The secondary embodied controller, consisting only of the robot body and linear feedback weights, handles the dynamics, i.e., it tunes the primary control signals to result in smooth and stable trajectories. In this task, only the desired end-effector trajectories are known, not the control signals required to generate them.

Thus far, in physical reservoir computing, embodied or morphological computation has always been exploited using supervised learning techniques. This implies that the target motor signals have to be known (e.g., determined using evolutionary techniques as in our own previous work) and fixed. However, in compliant robotics, it is important that the controller can adapt to variability in its surroundings as well as to changes of its own body. From this point of view, a reward-modulated approach is much more suitable. We demonstrate how these systems can be effectively trained in an entirely online manner.

2. Methods

In this section, we introduce the basic learning rule used throughout the paper in the context of neural networks, and we discuss additional changes to the rule to make it more suitable for the targeted application in partially embodied control of a tensegrity robot.

Throughout this work, we will employ the term *observations* instead of state or network activity, in order to emphasize that the learning rule is also applied in a more general context than neural networks.

2.1. Hebbian Learning in Analog Recurrent Neural Networks

Hebbian plasticity is a biologically plausible learning methodology for neural networks. A learning rule is called Hebbian if it modifies the weights between a set of presynaptic neurons $\boldsymbol{x}$ and postsynaptic neurons $\boldsymbol{y}$ as a function of their joint activity. Although Hebb (1949) did not provide a precise mathematical formulation of his postulate, a relatively general form can be written as:

$$\Delta W_{\text{Hebb}} = f(\boldsymbol{X}, \boldsymbol{Y}). \tag{1}$$

Note that we have used capital $\boldsymbol{X}$ and $\boldsymbol{Y}$[1] to indicate that the weight updates in the learning rule can depend on multiple time steps, i.e., the history of the pre- and postsynaptic neuron activations.

[1] We use the notation x to denote a scalar, $\boldsymbol{x}$ for a vector, and $\boldsymbol{X}$ for a matrix. In general $\boldsymbol{x}_{\text{i}}$ is the ith row of $\boldsymbol{X}$ and x_i is the ith element of $\boldsymbol{x}$.

To apply Hebbian theory in a reinforcement learning setting, we have to introduce the notion of a reward r into the learning rule. Indeed, reinforcement learning aims at making behavior that optimizes the reward more likely to happen. However, learning new behaviors necessitates another tool, namely exploration. We use noise $\boldsymbol{z}$ injected at the postsynaptic neurons for exploration.

If the exploratory noise causes an improvement in behavior, this will result in a higher reward (and vice versa). A basic learning rule based on this idea is:

$$\Delta \boldsymbol{W} = r\boldsymbol{z}\boldsymbol{x}^T. \tag{2}$$

Note that the postsynaptic neuron activations $\boldsymbol{y}$ are only indirectly considered in this weight update: the noise $\boldsymbol{z}$ can be viewed as a cause for variations in $\boldsymbol{y}$, and could be computed from the expected and noisy postsynaptic activations.

However, this rule suffers from a number of basic flaws. First, credit is only assigned to the exploratory noise that was inserted in the same time step that the reward was received. For the learning rule to be able to credit both past and present exploration, some efficient notion of memory of the relationship between exploration noise and the presynaptic neuron states needs to be present. This can be achieved by computing the covariance between the exploration and the presynaptic neuron states throughout multiple time steps. To this end, we will apply the rule on a trial-by-trial basis. Second, we note that in its current form, any significant bias of the reward r will cause unfavorable results. The solution to this is to predict the reward and subtract this from the obtained reward, resulting in a learning rule of the form:

$$\Delta \boldsymbol{W} = \alpha(r - \bar{r})\boldsymbol{Z}^T\boldsymbol{X}, \tag{3}$$

in which $\bar{r}$ is the predicted reward and where we have added a learning rate parameter α. The matrices $\boldsymbol{X}$ and $\boldsymbol{Z}$ contain the presynaptic neuron states and the exploratory noise throughout the trial, respectively.

The predicted reward is sometimes ambiguously referred to as the (short term) average reward. More precisely, it is the average (or expected) reward when noise is present in the system. As we will demonstrate, the average reward is typically highly dependent on the noise level of the system. The learning rule therefore optimizes the expected reward while noise is present in the system (i.e., $\max \mathbb{E}[r|\boldsymbol{z}]$), under the assumption that this also optimizes the performance when the exploration noise is removed (i.e., $\max \mathbb{E}[r|\boldsymbol{z} = 0]$).

Although RMH learning is stable in practice, it is possible to constrain the norm of the weights. This can be useful to do for practical reasons. In a robotics application, for example, this would allow for limiting the required feedback gain and thus the required motor power. In the Appendix, we show that in doing so, the resulting learning rule very closely resembles Sanger's rule (Sanger, 1989).

2.2. Decorrelated Learning Rule for Robotics Experiments

In partially embodied control, the dynamics of the robot body are used directly as a computational resource. In our RMH learning setup, this is equivalent to replacing part of the neural network by the robot body, which receives inputs from the remaining neurons. The observations, i.e., the sensor readouts, are fed back into the neurons. The training procedure is now heavily constrained, as it can only adapt synaptic weights of the remaining neurons, whereas the part of the network that is replaced by the body remains unchanged. Nonetheless, the RMH learning rule remains applicable, but the observations $\boldsymbol{x}$ and the noise $\boldsymbol{z}$ now include the sensor readouts and motor actuation noise, respectively.

Although this situation is similar to RMH learning in neural networks, it differs in the fact that most of the physical state of the robot remains hidden to the observer and the number of observable signals that can be fed into the trainable neural network is relatively small. Furthermore, the dynamics of the observed variables tend to be highly correlated. For example, stiffening the structure typically causes an increase in all sensor values.

The RMH learning relies on the varying influence of exploration noise on the observed variables. As we will show, a common influence (increase or decrease) of the noise on all observed variables reduces the effectiveness of the learning rule. A simple approach to overcome this issue is to decorrelate the observations. In Caluwaerts et al. (2012) (see Appendix), we showed that a decorrelation layer that uses Sanger's rule (Sanger, 1989) offers a biologically plausible solution for this. In this work, we take a more pragmatic approach and decorrelate $\boldsymbol{X}$ on a trial-by-trial basis, using the Moore–Penrose pseudoinverse. The resulting learning rule is given by:

$$\Delta \boldsymbol{W} = \alpha(r - \bar{r})\boldsymbol{Z}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}, \tag{4}$$

with λ acting as a regularization parameter determining the strength of the decorrelation. A slight variant of this rule is:

$$\Delta \boldsymbol{W} = \alpha\mathcal{H}(r - \bar{r})\boldsymbol{Z}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}, \tag{5}$$

where $\mathcal{H}(\cdot)$ represents the Heaviside step function. In this variant of the learning rule, weight updates only occur when the observed reward is better than expected. This is not strictly necessary, but we found that it slightly improved our results.

This learning rule is similar to ridge regression, the difference being that the algorithm will try to reproduce the noise $\boldsymbol{Z}$ (instead of a desired output) proportionally to the reward with injected noise, relative to the expected reward. We used a large regularization parameter ($\lambda = 1$), which results in only a minor decorrelation of the sensor variables, yet is enough to allow for efficient learning. A high-regularization parameter allows for simple covariance estimators, in case a more biologically plausible version of the rule is desired.

3. Experiments

3.1. Neural Network Experimental Setup

Before presenting our results in a robotics context, we first study RMH plasticity in discrete time recurrent neural networks with hyperbolic tangent activation functions. They receive input, which we denote $\boldsymbol{U}$ and a readout function provides observations of the network state. Additionally, exploration noise $\boldsymbol{Z}$ is injected into the network. The network update equation is given by:

$$\boldsymbol{x}[k+1] = \tanh(\boldsymbol{W}\boldsymbol{x}[k] + \boldsymbol{W}_{\text{in}}\boldsymbol{u}[k+1] + \boldsymbol{z}[k+1]). \tag{6}$$

For each of our neural network experiments, the network is initialized according to the reservoir computing approach (Verstraeten et al., 2007). This implies that we initialized the weights randomly (i.i.d. standard normally distributed samples) and then tune the network dynamics to a useful regime. This is achieved by rescaling the weight matrix $\boldsymbol{W}$ such that its spectral radius – the largest amongst the absolute values of its eigenvalues – is such that the learning converges. For the experiments we will describe, we obtained good performance for initial spectral radii in [0.80, 1.2], i.e., stable or almost stable networks. This differs from related approaches, such as Legenstein et al. (2010) and Hoerzer et al. (2012), where initially chaotic networks are used. The input weight matrix $\boldsymbol{W}_{\text{in}}$ was sparsely initialized (20% non-zero elements) with i.i.d. normally distributed values with standard deviation (SD) 0.05. All networks contained 100 neurons.

Figure 1 shows our learning setup for neural networks. The neural network to be trained is the central element. A reward is provided after a trial based on the network observations throughout that trial. A trial is defined as the number of time steps in which the network tries to perform a task of interest. In parallel to the network, a reward prediction system estimates the expected reward based on the network inputs. The RMH learning rule finally combines information from the network state, exploration noise, reward, and estimated expected reward to compute an update $\Delta \boldsymbol{W}$ of the network weights.

3.2. Neural Network Experiments

The networks used for the three tasks described below are shown in **Figure 2**. They only differ in the way observed outputs are generated and in the subset of weights that can be modified by the learning rule. In the first two networks, two neurons are randomly selected as output-generating neurons, and the output is computed as the sum of their states. The third network has three output-generating neurons and has an output equal to the product of these neurons' states.

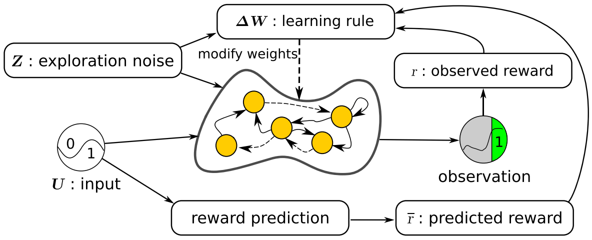

$\Delta \boldsymbol{W}$: learning rule
$\boldsymbol{Z}$: exploration noise
modify weights
r : observed reward
0
1
$\boldsymbol{U}$: input
1
observation
reward prediction
$\bar{r}$: predicted reward

FIGURE 1 | Overview of the learning setup for recurrent neural networks. The initially random recurrent neural network receives the inputs $\boldsymbol{U}$ and the exploration noise $\boldsymbol{Z}$. The state of the postsynaptic neurons is computed by applying the hyperbolic tangent function to the sum of the inputs, the noise, and the weighted sum of the presynaptic neurons. Observations are made of the state of the network and after every trial (fixed number of time steps), a reward is computed, based on the observations made during the last trial. In parallel, a simple reward prediction network predicts the expected reward for the given input. The learning rule then updates the weights between the presynaptic and postsynaptic neurons, by using the reward, the expected reward, the exploration noise, and the states of the presynaptic neurons.

In all networks, the weights to and from the output neurons are fixed and recurrent. This prevents the learning algorithm from generating solutions in which the observation neurons become a pure output layer, which does not influence the state of the rest of the network. In the networks for tasks 1 and 3, all other internal weights are modifiable. In task 2, the training is further restricted by fixing the input weights for half of the remaining neurons. This means that about half of the network is a random recurrent neural network. In a neural or neurorobotics context, we can see this as a rudimentary model for a trainable network interacting with an untrained dynamical system, such as another brain area or a physical body, e.g., the partially embodied control of a robot arm with a neural network.

We purposely chose to have different and unconventional tasks and setups, to display the wide applicability of reward-modulated Hebbian learning. In what follows, we describe the three neural network tasks in more detail. We first consider problems with discrete input spaces. More precisely, we solve the 2-bit delayed XOR problem and a 3-bit decoder task. Our third example has a continuous input space and a more complex readout function.

3.2.1. Task 1: Proof-of-Principle

3.2.1.1. Inputs

The input signal for this task represents a single bit stream. A zero bit is coded as the negative half period of a sine wave, a one bit as the positive half. For each trial, we randomly select one of the four possible 2-bit input sequences.

3.2.1.2. Desired output

The neural network has to compute the so-called 2-bit delayed XOR task, i.e., the exclusive OR function applied to the last two bits of its input stream, represented as binary values. More concretely, the output of the network should be as close as possible to plus one or minus one during the last half of the second bit.

The XOR task is a common test or benchmark, because the patterns are not linearly separable. A linear network cannot obtain optimal performance for all inputs simultaneously. Therefore, this task is a simple test to verify if the learning rule can exploit the non-linear effects of the network. The task also requires the network to remember a specific part of the input, while ignoring inputs that occurred more than one bit length in the past.

3.2.1.3. Neural network structure

The observations are computed by adding the states of two output neurons, which have fixed (i.e., untrained) incoming and outgoing connections. All other weights are trainable.

3.2.1.4. Reward function

Throughout this manuscript, we use different reward functions. The main reasons for this is that some reward functions are more appropriate for a specific task and to show that the learning rule does not depend on a specific reward function. For the results presented for the 2-bit XOR task, we used minus the mean squared

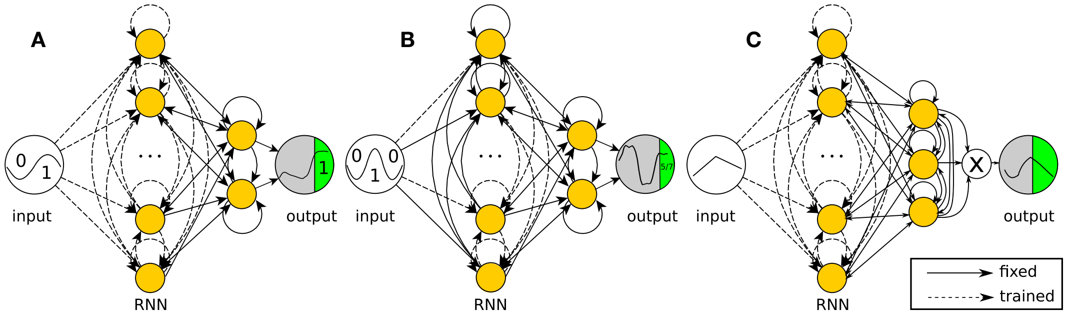


FIGURE 2 | Network structures for the recurrent neural network tasks. The networks are simulated in discrete time with hyperbolic tangent neurons (yellow nodes). Full lines are fixed connections, while dashed lines are trained. The reward is evaluated at the output neuron over the green period of time (one reward per trial). **(A)** Task 1 (2-bit delayed XOR): the output equals the sum of two neurons, but only the internal connections can be modified by the algorithm. **(B)** Task 2 (3-bit decoder): the output equals the sum of two neurons as in the XOR task, but now only half of the internal weights can be modified. **(C)** Task 3 (continuous input task): the output equals the product of three neurons. The network has to reproduce the reversed input from the first 5 time steps of each trial.

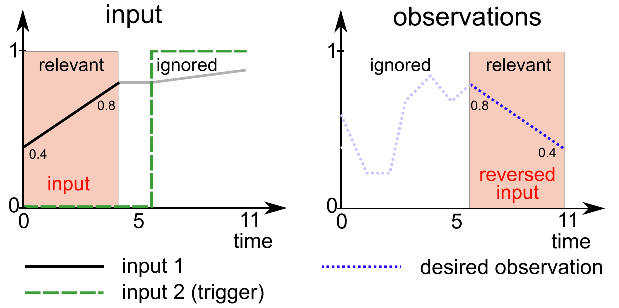


FIGURE 3 | Overview of task 3: the network has to reproduce part of the first input (black line) in reverse at the end of the trial (dashed blue line). More precisely, the first 5 steps of the first input are to be reproduced in reverse at the end of the trial (12 time steps total). The input space consists of a straight line originating and ending in [0, 1]. A second input indicates when the network has to start producing the desired observations.

hinge loss, as the hinge loss is a more appropriate reward function for a binary classification task:

$$r_n^{2\text{bit}} = \frac{-1}{5}\sum_{k=15}^{19} \max\left(0,\ 1 - t_n[k]\left(x_0^{\text{o}}[20n+k] + x_1^{\text{o}}[20n+k]\right)\right)^2, \tag{7}$$

where n indicates the number of the current trial, $\boldsymbol{x}^{\mathbf{o}}$ are the neurons that generate the observations and $t_n[k]$ are the desired observations.

3.2.1.5. *Prediction of the expected reward*

Estimating the expected reward is trivial in the case of a modest number of different inputs. For the results presented here, we averaged the last 50 rewards per input sequence.

3.2.2. Task 2: Partially Embodied Computation

3.2.2.1. *Inputs*

For this task, the input is the same as for the previous task. For each trial, we now randomly select one of eight possible 3-bit input sequences.

3.2.2.2. *Desired outputs*

The network now has to act as a 3-bit digital-to-analog decoder, i.e., it has to produce one of eight equidistant analog values in the range [−1, 1], corresponding to the decimal interpretation of the last three encoded bits it received. Similar to the previous task, the desired value has to be present on the output during the second half of the third bit. This task is more complex and non-linear than the previous one and it requires more memory as well.

3.2.2.3. *Neural network structure*

The output generation is identical to task one. However, this time only half of the internal weights are trainable.

3.2.2.4. *Reward function*

For the 3-bit decoder task, the reward value $r^{3\text{bit}}$ is defined as minus the mean squared error of the observations during the last five time steps of a trial:

$$r_n^{3\text{bit}} = \frac{-1}{5}\sum_{k=25}^{29}\left(t_n[k] - x_0^{\text{o}}\left[30n+k\right] - x_1^{\text{o}}[30n+k]\right)^2. \tag{8}$$

3.2.2.5. *Prediction of the expected reward*

The rewards were estimated in the same way as in the previous task.

3.2.3. Task 3: Non-Linear Observation Function

The task at hand is to reproduce part of the input in reverse after a delay (see **Figure 3**).

3.2.3.1. *Inputs*

The network receives two input signals. The first input signal consists of sequences of 12 time steps per trial. The first 5 of these steps form a linear segment between two values, which are sampled with uniform probability form [0, 1] for each trail. The final value is held constant for two more time steps. The final 5 time steps again form a linear segment that is obtained by connecting the last value from the first 5 steps with a third random sample from [0, 1].

Because the trials are fed into the system one by one, it is not clear to the network when a trial starts. The second input, a binary signal, is used to inform the network when it has to generate the desired output.

3.2.3.2. Desired outputs

The network must learn to recall the input during the first 5 steps and reproduce them in reverse order during the last 5 steps of the trial. The system must ignore the remaining 7 input samples.

3.2.3.3. Neural network structure

The observations are computed by multiplying the states of three internal neurons, which have fixed (i.e., untrained) incoming and outgoing connections. All the other weights are trainable.

3.2.3.4. Reward function

The reward function used here is minus the mean absolute error of the observations:

$$r_n^{\text{cont}} = \frac{-1}{5}\sum_{k=0}^{4} \left| u\left[12n+k\right] - \prod_{j=0}^{2} x_j^{\text{o}}\left[12n+11-k\right] \right|. \tag{9}$$

3.2.3.5. Prediction of the expected reward

The expected reward $\bar{r}$ estimates the performance of the system given the noise level σ. Furthermore, the reward is input dependent, therefore $\bar{r}$ estimates the following quantity:

$$\bar{r} = \mathbb{E}[r|\boldsymbol{u}, \sigma]. \tag{10}$$

Various algorithms can be used to estimate this quantity. We employed the well known recursive least squares algorithm (Kailath et al., 2000) to learn a simple online estimator of this quantity, which we applied to the input sequences $\boldsymbol{u}$ and the network state $\boldsymbol{x}$ at the end of a trial.

3.3. The Tensegrity Robot

The general setup of our simulated tensegrity robot control problem is shown in **Figure 4**. It is similar to the neural network setup represented in **Figure 1**, but the entire recurrent neural network has been replaced with the simulated tensegrity robot. As a result, the only remaining trainable weights are those of a simple linear feedback $\boldsymbol{W}$, projecting the output to the input.

The tensegrity structure used for our experiments has four struts and is shown in **Figure 5**. It is based on the standard three strut tensegrity prism (Pugh, 1976) to which a shorter rod has been added that acts as a compliant end-effector. The bottom three nodes of the original prism have been fixed through ball-joints. The resulting structure has seventeen $k = 20\,\text{N}{\cdot}\text{m}^{-1}$ springs, 14 of which are actuated (the lengths of the other three bottom springs are fixed). The controller time step was 50 ms and gravity was not modeled.

Instead of observing the state of the neurons, we measure the spring forces:

$$\begin{aligned} x_i &= f_i & (11)\\ &= \max(k(l_i - l_i^0), 0). & (12)\end{aligned}$$

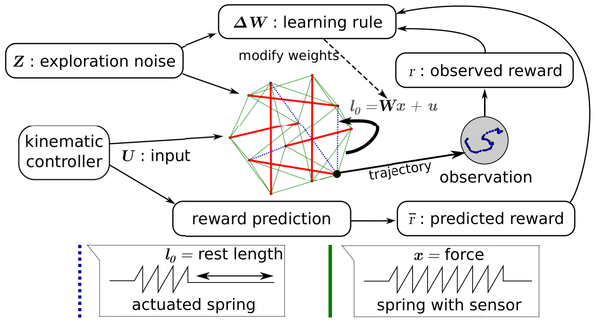


FIGURE 4 | Overview of the way the learning rule is applied to compliant tensegrity structures. The setup is similar to the recurrent neural network setup of **Figure 1**. The neural network has been replaced by the combination of the compliant robot body and the neural linear feedback weights. It now receives input from the kinematic controller. Force sensors on the springs act as presynaptic neurons for the trained weights and the actuator signals correspond to the postsynaptic neurons. The learning rule adapts the feedback weights from the force sensors to the motor signals. The observations used for reward computation are based on the trajectories of an end-effector.

Actuators changing the equilibrium lengths of the springs replace the postsynaptic neurons and motor babbling takes on the role of the exploration noise:

$$\boldsymbol{l}^0 = \boldsymbol{l}^{\text{init}} + \boldsymbol{W}\boldsymbol{x} + \boldsymbol{u} + \boldsymbol{z}. \tag{13}$$

All tensegrity experiments were performed in our tensegrity simulator, which is based on an Euler–Lagrange formulation of the tensegrity dynamics (Skelton and de Oliveira, 2009, chapter 5). For more details on the simulation setup, we refer to Caluwaerts et al. (2012).

3.4. Hierarchical End-Effector Control for the Tensegrity Robot

The task we consider is writing characters with the top node of the rod suspended in the tensegrity structure. More precisely, the node has to trace letters in a horizontal (XY) plane. The characters were taken from UCI Character Trajectories Data Set (Bache and Lichman, 2013), integrated and then subsampled and rescaled.

The robot is controlled by combining a feed forward kinematic controller and a learned static linear feedback controller. The kinematic controller provides the input signals $\boldsymbol{u}$ in equation (13). We sampled 100 random spring lengths to create a set of configurations for the kinematic controller. To write a character, the kinematic controller selects a combination of spring lengths that move the end-effector as close as possible to the desired position when the structure is in equilibrium.

The reward function used for the next experiments tries to bring the end-effector close to the desired trajectory:

$$r^{\text{traj}} = \frac{-1}{s}\sum_{k=0}^{s-1} \max(\|\boldsymbol{n}[k] - \boldsymbol{c}[k]\| - 0.01,\ 0), \tag{14}$$

where s is the number of steps required to write the current character, $\boldsymbol{c}[k]$ the vector containing the target position at time k

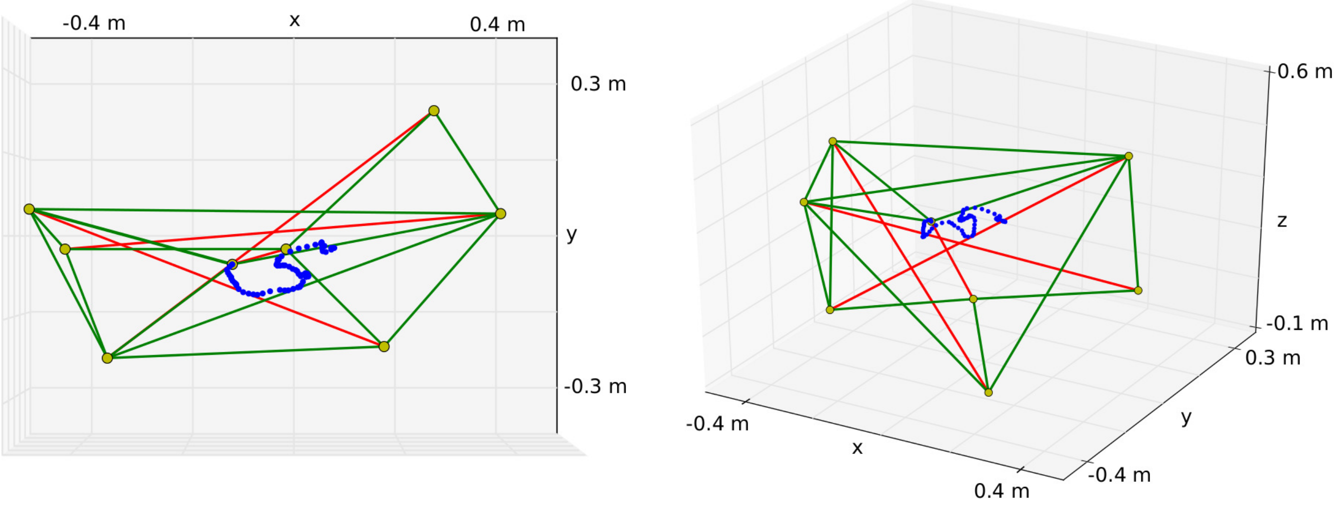


FIGURE 5 | Tensegrity structure used for the experiments. The top node of the center rod is used as an end-effector to draw in the XY plane. In this example, the robot draws an "S" as can be seen on the left. The right figure shows another perspective to demonstrate that the reward does not depend on the vertical position.

(relative to the beginning of the trial) and $\boldsymbol{n}[k]$ the position in the XY plane of the end-effector at time k. This reward function will cause the learning rule to stop improving a feedback controller w.r.t. a point on the trajectory in case the end-effector is within 1 cm of the target position.

3.5. Robustness Against Failures

To demonstrate the robustness of the controllers as well as a more practical application of the learning rule, we simulated various actuator failures. In this case, we used a more realistic feed forward controller. Again starting from a simple kinematic controller, we now optimized the inputs $\boldsymbol{u}$ in equation (13) at each time step using a basic exploration method. More precisely, a small amount of noise $\boldsymbol{z}$ is injected at each time step and the change in expected reward is observed. If an improvement of the expected reward is observed after a trial, we reproduce the noise in the feed forward controller $\boldsymbol{u} = \boldsymbol{u}_{kin} + \boldsymbol{u}_{expl}$ by using $\Delta \boldsymbol{U}_{\text{expl}} = \boldsymbol{Z}$ when the expected reward (of the trial) improved and $\Delta \boldsymbol{U}_{\text{expl}} = 0$ when it did not. In the previous equations, $\boldsymbol{u}_{kin}$ refers to the original kinematic controller discussed in the previous section.

In this setup, we now simulate actuator failures by resetting one or more actuators to the original kinematic controller instead of the optimized ones, i.e., $\boldsymbol{u} = \boldsymbol{u}_{\text{kin}}$. At the same time, the kinematic controller is no longer optimized, and instead the learning rule starts learning a set of feedback weights to compensate for the actuator failure.

4. Results

4.1. Neural Network Experiments

We first demonstrate the learning rule's capabilities using the neural network tasks. For these experiments, we always chose the noise level to be $\sigma = 0.05$ and set the learning rate to be as high as possible, without making the networks diverge. In the context of the basic learning rule [equation (3)] this was $\alpha = 0.005$, whereas the decorrelated version [equation (4)] allowed a much higher learning rate $\alpha = 0.5$. The initial spectral radius was chosen to be 0.95.

Figure 6 shows the evolution of the reward and the spectral radius for the discrete input tasks (tasks 1 and 2). We performed 10 runs with different random initializations and input sequences. For both tasks, and for every run, we observe that the network indeed learns to solve the task almost perfectly, as the rewards converge to their maximal value of 0.

These experiments also show how the spectral radius evolves as the average reward increases. In the case of the 2-bit delayed XOR task, every run of the algorithm for different initial random weights resulted in a final spectral radius of approximately 1.2, which indicates that the learning rule tunes the memory of the network. In the case of task 2, the learning rule is only allowed to modify half of the internal weights of the recurrent neural network. The network structure can be considered as a model for partially embodied computation, i.e., we replace part of the original trainable network by a fixed one, which acts as a dummy for a physical body. Nonetheless, the learning rule manages to tune the network dynamics to have a spectral radius close to 1.05 after 300,000 trials, which eventually converges to 1.10 (not shown), thus exhibiting the necessary memory.

We compared our results to an approach in which the trainable weights are updated based on an estimate of the noise, instead of using the real noise, similar to the EH rule described in Legenstein et al. (2010). The noise is estimated as the difference between the neural input $\boldsymbol{a}$, and the expected neural input $\bar{\boldsymbol{a}}$, $\boldsymbol{z}_{\text{estim}} = \boldsymbol{a} - \bar{\boldsymbol{a}}$. The expected neural input $\bar{\boldsymbol{a}}$ is simply an exponentially weighted moving average of $\boldsymbol{a}$ with a smoothing factor of 0.8. However, we found that this approach performed severely worse on the tasks we considered. A typical example of a 300 neuron network[2] trained on the 3-bit decoder task is shown in **Figure 7**. The top panel plots the evolution of the reward during the training. It shows that not

[2] The experimental results using only 100 neurons are qualitatively similar, but less pronounced.

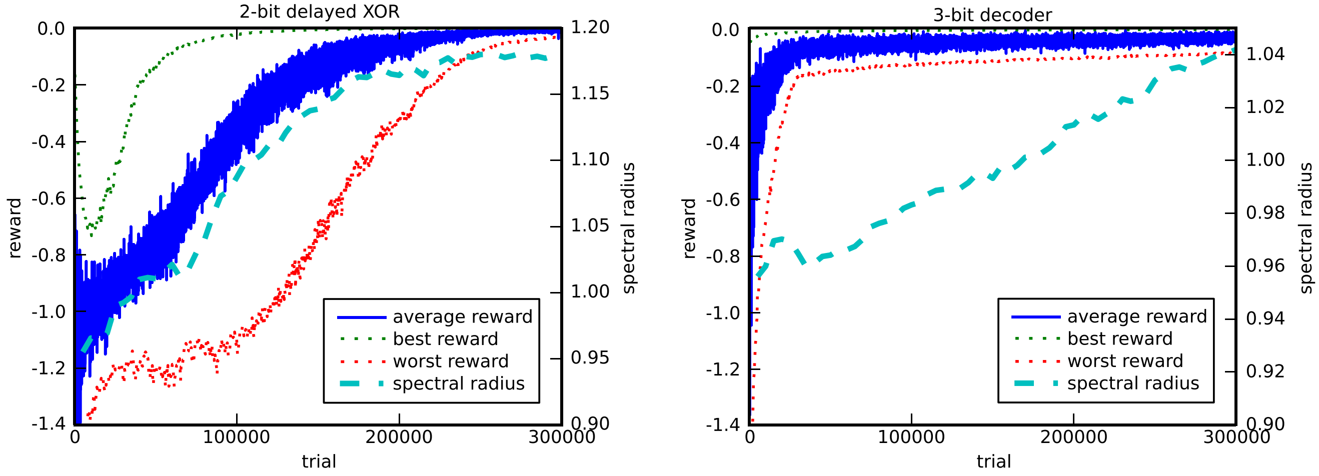


FIGURE 6 | Evolution of the reward and the spectral radius for the 2-bit delayed XOR task (left) and the 3-bit decoder task (right). The plots show the average, best, and worst rewards and the average spectral radius out of 10 runs. Each run has different random initializations and input sequences.

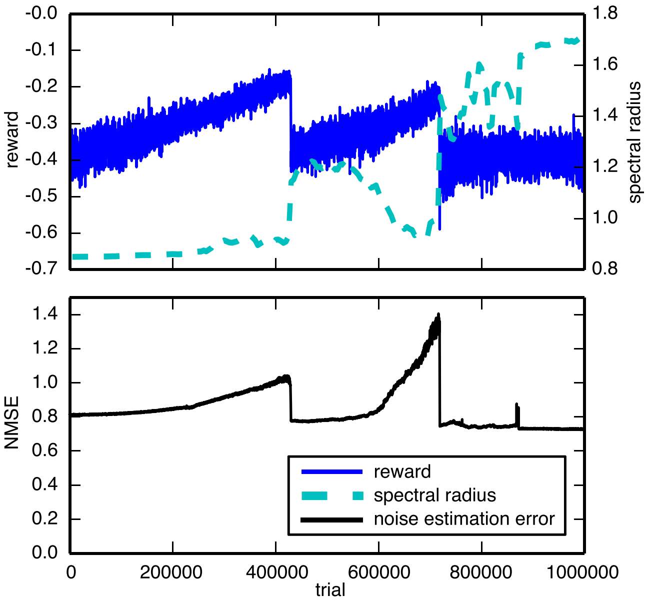


FIGURE 7 | Evolution of the reward and spectral radius (top) and the noise estimation NMSE (bottom) for the 3-bit decoder task, using a learning rule that uses an estimate of the exploration noise instead of the real noise. The network consisted of 300 neurons, instead of 100, and was trained over a longer period than the experiments shown in **Figure 6**.

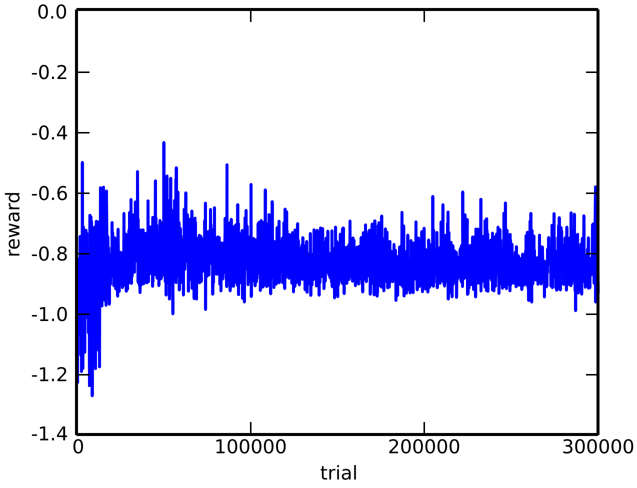


FIGURE 8 | Evolution of the reward for the delayed XOR task, using CMA-ES. The task was simplified by removing all forms of stochasticity.

only is training much slower but also goes through a couple of bifurcations from which it is eventually unable to recover. The bottom panel shows why it is difficult to train the networks using this rule. We see that the noise estimation error slowly increases as the training continues. For this learning rule to work well, we require a good estimate of the noise throughout the entire learning process.

As a second comparison, we evaluated the performance of the covariance matrix adaptation evolution strategy (CMA-ES) (Hansen and Ostermeier, 2001), one of the most popular evolutionary algorithms. The algorithm is used on the 2-bit delayed XOR task. To make the task a bit simpler, we removed all sources of stochasticity (noise, initial neuron state), but apart from this, the setup is completely identical (identical network architecture, same initialization). The objective function to be maximized is the average reward across the four different possible inputs. **Figure 8** shows the evolution of the reward during the first 300,000 trials. Comparing this to left panel of **Figure 6**, we can see that, although the RMH rule is able to find a good solution after 300,000 trials, CMA-ES is still nowhere near converging to a solution. The likely explanation for this is that, because the search space is so huge (about 10,000 dimensions), sample-based approaches like this one require an unfeasible number of samples.

Figure 9 shows in more detail what the embodied network of task 2 has learned, by overlaying the network output during 50 random orderings of the input sequences. Note that the classification result must only be available at the output during 5 time steps at the end of each trail (indicated by red crosses in the figure).

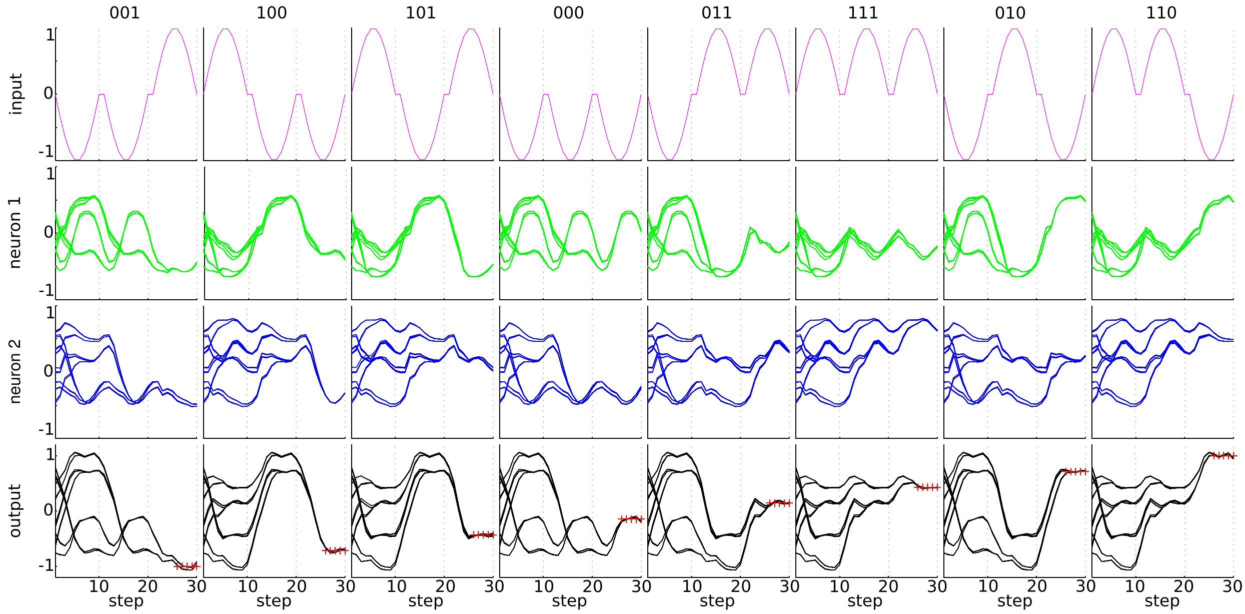


FIGURE 9 | State trajectories for the 3-bit classification task. The top row shows the eight input sequences. The next two rows show the state trajectories of the two output neurons, which are summed to compute the observations for the given input sequence of the top row. The bottom row shows the observations (the sum of the two middle rows). The desired observation at the end of the trial is indicated by red crosses. The plots were generated by overlaying 50 random orderings of the input sequences.

As trails follow each other continuously, the variability of the state trajectories is due to the different initial states. Clearly, the network has learned the correct time window, as the classification result is available at the right time and only depends on the two previous bits and not on older inputs. We can identify a number of separate trajectories that keep track of the possible outcomes. It can be seen that within a trial, each additional bit that is offered at the input reduces the number of possible states of the system by half. Interestingly, the two neurons that generate the observations have different state trajectories, because the learning rule only quantifies the performance based on the observations, without directly enforcing a specific behavior of the neurons responsible for the observations.

Although the network was trained without any noise on the input signals, the resulting behavior is robust against such noise. In **Figure 10**, we show the behavior of the network when input noise is present. This plot was generated by first applying k-means clustering on the trajectories and then estimating the variance of each centroid. Shown are the various centroids and the SD of each. We see that the network is robust against high amounts of noise on the input data (σ up to 0.5), as the original trajectories are maintained.

The same noise robustness can be observed on the last and most elaborate task. **Figure 11** visualizes the rewards during testing for various state noise levels, using 100,000 random input trials per noise level. The noise was added to the internal neurons of the network, but not to the three neurons, which generate the observations. Each graph in the top panel shows the average rewards of the trained networks, across the whole spectrum of possible input sequences for a given level of input noise (increasing from left to right). The bottom panel shows the reward distribution, averaged across the different input trials, for each noise level. We see that without noise, the average reward remains close to its optimal value of 0 for most input patterns, although some regions of the input pattern space seem to be slightly more difficult. This demonstrates that the learning rule also works, although less perfectly than in the previous cases, when the relation between the internal states in the network and the way they are translated into actions and rewards is highly non-linear and when the input patterns do not fall into discrete categories. As noise levels increase, the average reward decreases, but only slightly, again displaying the noise robustness of the trained networks.

Finally, we show the virtue of the decorrelation learning rule [equation (4)] by slightly modifying the setup of the 3-bit decoder task. Instead of using uncorrelated Gaussian noise with SD $\sigma = 0.05$ for exploration, we generate the noise by sampling from a Gaussian distribution with SD $\sigma = 0.035$. The mean of the distribution is in turn sampled from a Gaussian with identical SD and zero mean, but only once per trial. During a single trial, the mean noise value is kept constant. This sampling procedure is nearly equivalent to the original one, except for the fact that two samples within the same trial are now highly correlated.

Figure 12 compares the default and the decorrelation learning rules by plotting the average reward evolution for both the original task 2 setup and the setup with correlated noise. Again, we performed 10 runs with different random initializations and input sequences to generate each curve. In the left panel, we observe that in the context of uncorrelated noise, both learning rules give virtually identical results. In the case of correlated exploration noise though, the decorrelation learning rule does much better than the default one. The default RMH rule is able to learn at the

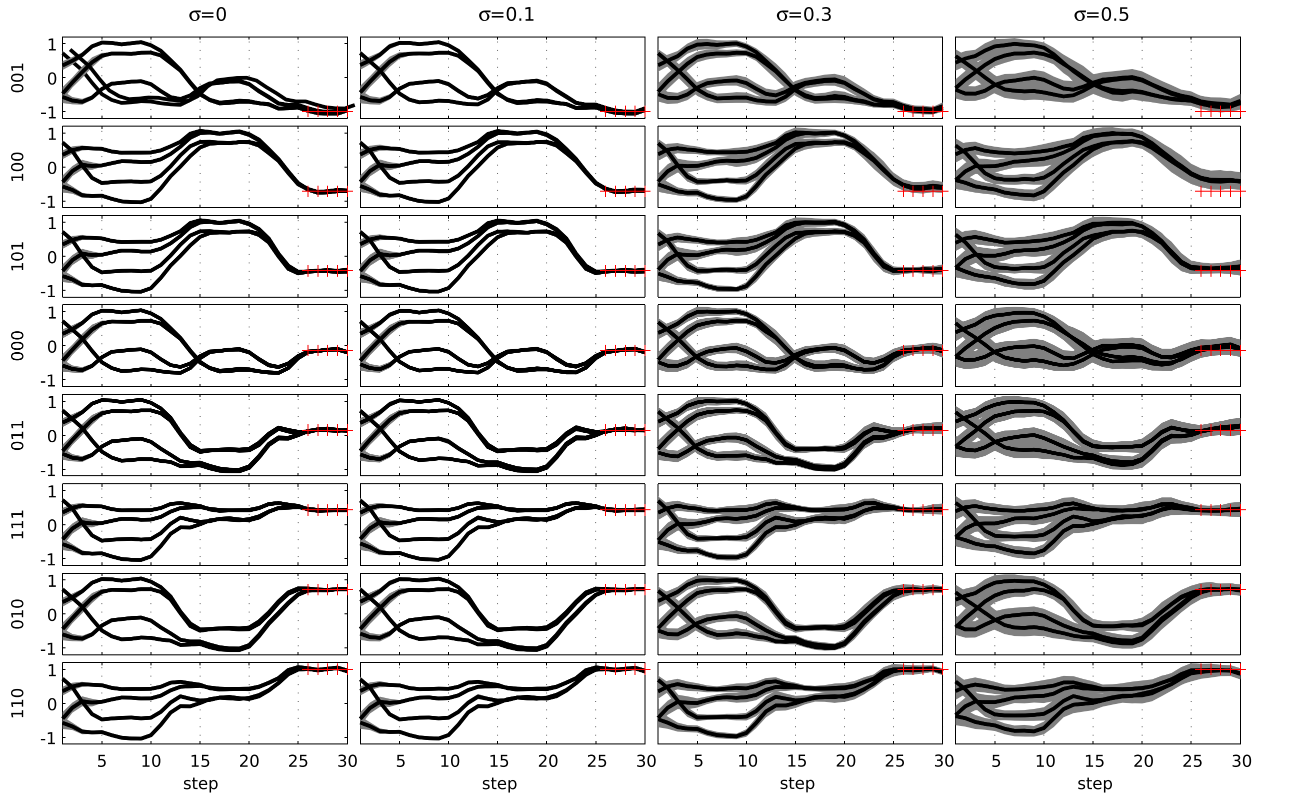


FIGURE 10 | Evaluation of a network trained for the 3-bit decoder task under the influence of input noise. The noise amplitude increased from left to right. The eight possible inputs are shown from top to bottom. The gray area indicates 1 SD around the observations for each of the different state trajectories (thick lines). The red crosses indicate the target observations at the end of the trial.

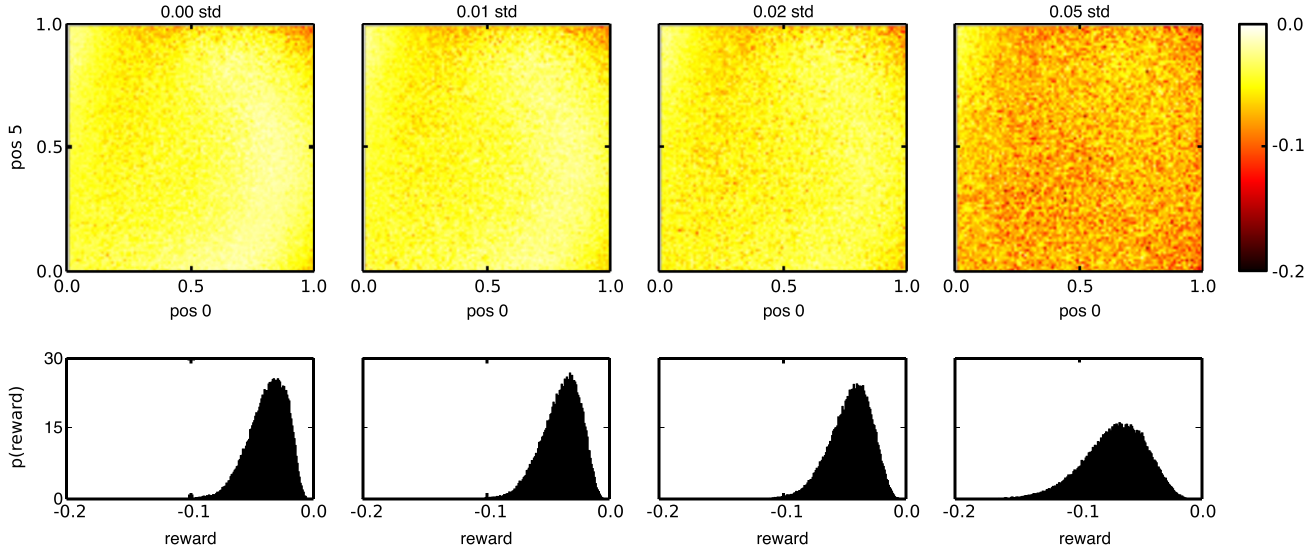


FIGURE 11 | Reward distribution for task 3 during testing with different state noise levels. (Top) average reward (negative mean absolute error) for the whole range of input combinations. The horizontal and vertical axes of each plot indicate the initial and final values of the linear segments that need to be reproduced, in reverse order, at the output. (Bottom) sample distribution of the rewards for all inputs. The amount of injected state noise increases from left to right.

start of a run, but quickly falls back and eventually settles on a suboptimal result. In contrast to this, the decorrelating version of the learning rule exhibits a very healthy learning curve and is only slightly affected by the fact that the exploratory noise is correlated.

4.2. Tensegrity Experiments

Having shown the applicability of reward-modulated Hebbian learning on different tasks, using diverse setups, we now move on the tensegrity robot experiment. In this experiment, only a set of feedback weights are trainable, i.e., all neurons in a neural network controller have been replaced by the robot body.

The left panel of **Figure 13** shows how the tensegrity robot performs when drawing characters between 20 and 68 time steps long (1–3.4 s). A different set of feedback weights was learned for each character; therefore, it is easy to predict the expected reward. To clarify, we estimated the expected reward for each character individually by averaging the rewards obtained during the previous 30 trials. As can be seen from the top row, the initial performance of the system with only the kinematic controller is very low, whereas the combination of both controllers, using our RMH learning rule [equation (5)], performs considerably better.

The plot on the right of **Figure 13** shows the learning curves for each character, indicating that for most characters good results were obtained after 1000–1500 trials, which would be equivalent to <1 h real robot time for most characters. It is possible to accelerate learning by further tuning the learning parameters. We used a conservative level of exploration noise ($\sigma = 5$ mm) and a learning rate $\alpha = 1$, which consistently resulted in stable feedback controllers. The learning rule did not achieve the same final reward for all characters (e.g., the “m”). This is due to physical limitations enforced on the motor commands.

Finally, we simulate actuator failures. The results of these experiments are presented in **Figure 14**. As could be expected, the performance immediately drops significantly after each failure. By applying the RMH learning rule to the feedback controller, the system is able to recover from the various failures. To investigate the stability of the learning rule, each experiment was performed 30 times, with similar results.

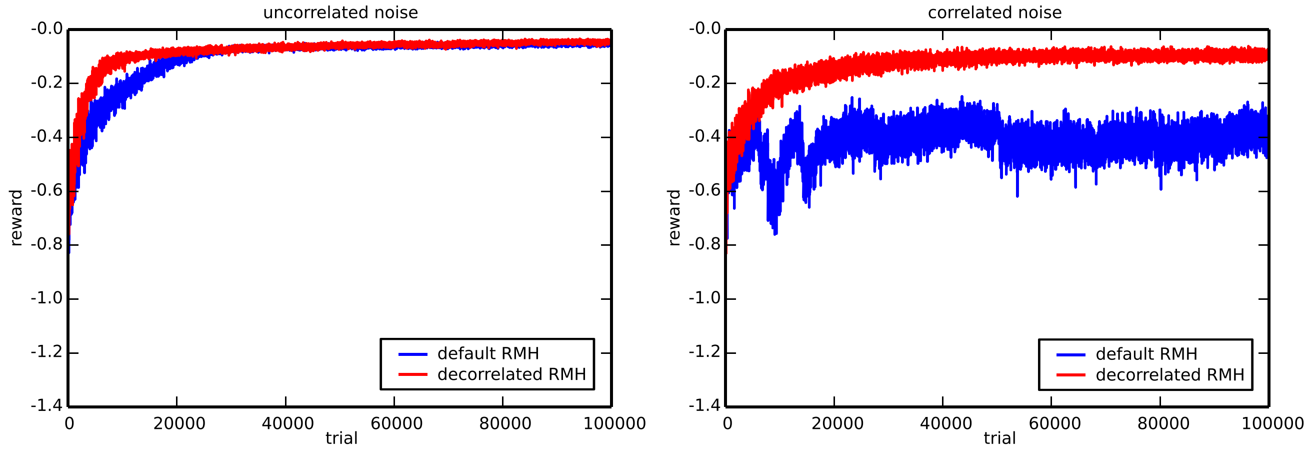


FIGURE 12 | Comparison of the default and the decorrelated RMH learning rules. The evolution of the average reward for the 3-bit decoder task is shown in the presence of uncorrelated (left) and correlated (right) noise. The plot is generated using ten runs with different random initializations and input sequences.

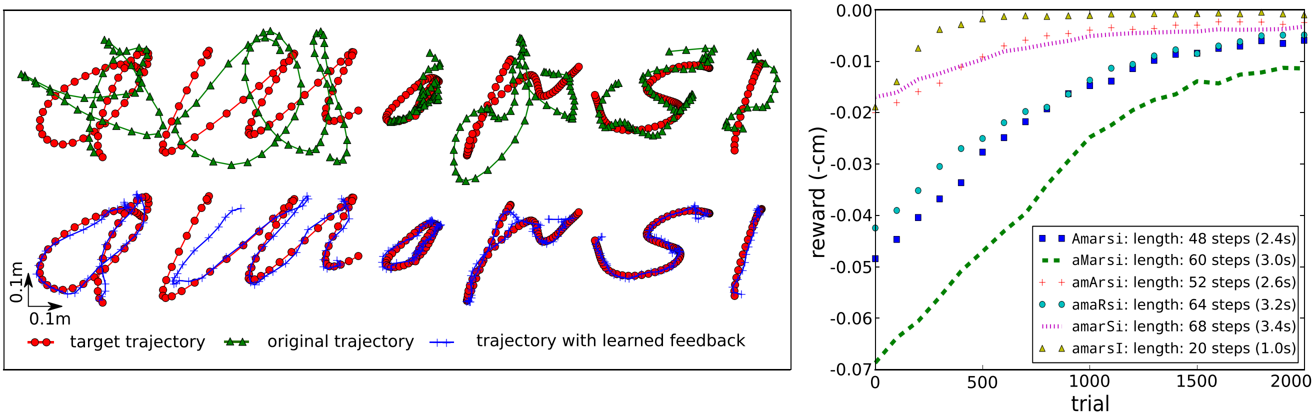


FIGURE 13 | Writing characters with a tensegrity end-effector. (Top left) characters drawn with only the kinematic feed forward controller active. (Bottom left) characters drawn with the kinematic feed forward controller and the learned feedback controller active. (Right) learning curves for the different characters. The legend indicates the length of a trial.

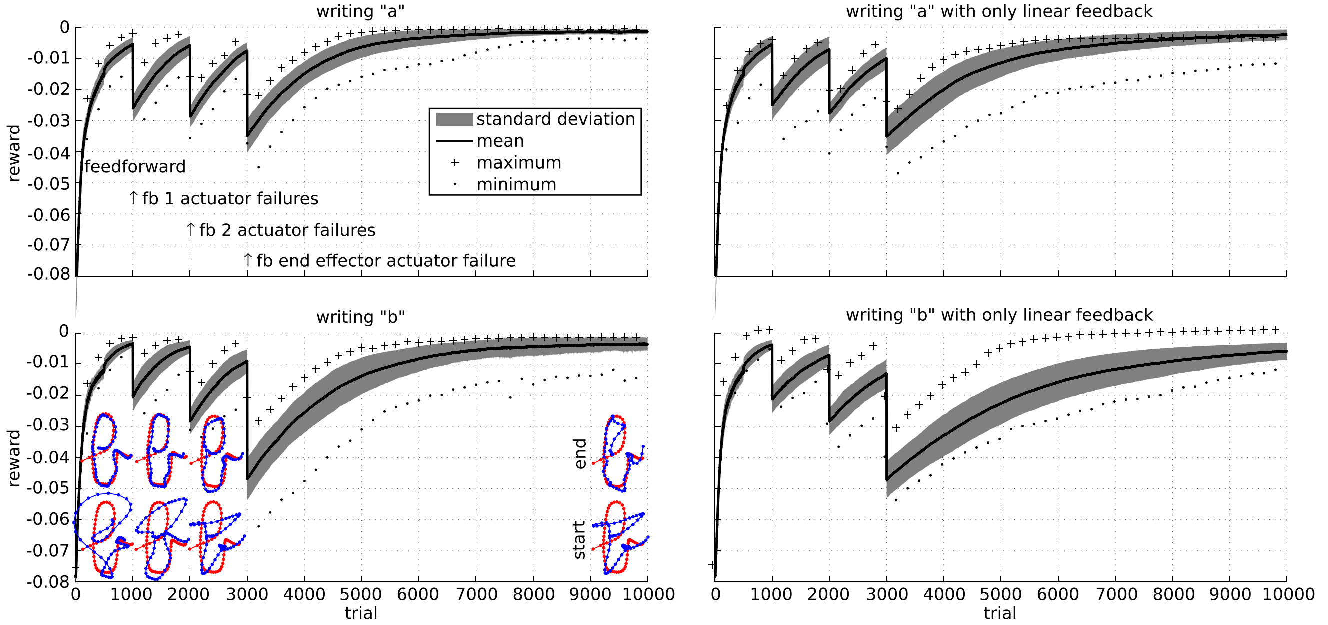


FIGURE 14 | Robustness of the learning rule for the writing task. Initially, a feed forward controller is optimized. We then simulate a failure by making a single actuator follow its initial trajectory from trial 1000 onward. At the same time, the learning rule starts learning a set of feedback weights to compensate for the actuator failure. Similarly, we simulate a two actuator failure after trial 2000. At time 3000, we simulate a failure of an actuator directly attached to the end-effector. The top row shows the results for writing an "a" character, while the bottom row shows the results for a "b." The left column show the results when the feedback includes the spring forces and the square of the spring forces, while the right column only includes the spring forces. The plots show the mean, maximum, minimum, and SD of the reward over 30 runs.

5. Discussion

Hebbian theory is a well-established approach to explain synaptic plasticity between neurons. Over the years, many variations of the basic learning rule have been developed. Each of these had a specific application, ranging from unsupervised feature extraction to reinforcement learning. In this work, we handled Hebbian-like learning rules in which the synaptic plasticity is based on the correlation of the presynaptic neurons and an exploratory noise signal. The plasticity was modulated by a reward signal, resulting in a learning method that maximizes the expected reward of a trial.

We showed that this kind of learning can be applied outside the scope of traditional neural networks, namely in embodied computation. While similar rules have already been presented, we focused on reward learning in constrained recurrent neural networks and compliant robots. The rationale for this is our belief that both can be seen as computational resources and can therefore benefit from similar learning techniques.

Our work builds upon Legenstein's (Legenstein et al., 2010), who considered simulated motor control tasks in combination with an instantaneous reward signal in an initially chaotic neural network. One significant difference with respect to our experimental setup is that Legenstein estimated the exploration noise as well as the expected reward. This allows for uncontrolled or unknown noise sources to be used, which adds to the biological plausibility of the method learning (Faisal et al., 2008). Covariance and noise-based rules have a strong biological foundation (Loewenstein and Seung, 2006; Soltani and Wang, 2006; Loewenstein, 2008). For example, it is well-known that neural networks in biology have intrinsic noise sources (Faisal et al., 2008), which could be used for learning (Maass, 2014). While this type of noise can sometimes be measured by external means (e.g., voltage clamps), a plasticity rule within the biological substrate cannot generally observe the noise signals, hence the importance of the noise estimator in Legenstein's rule. In this work, we considered this approach briefly, but observed unfavorable results. The noise estimation scheme used by Legenstein requires the input and network dynamics to be temporally stable on small time scales, which likely explains our observations. However, apart from its biological plausibility, such a scheme is unnecessary in our context, as significant uncontrolled noise sources seem unlikely in robotics. Finally, we remark that Legenstein's learning rule extends various earlier techniques with similar mathematical formulations (Fiete and Seung, 2006; Loewenstein and Seung, 2006; Loewenstein, 2008).

Another important difference of the learning rule that we considered with more biologically plausible alternatives, like the ones presented in Legenstein et al. (2008) and Soltoggio and Steil (2013), is that we employed trial-based learning. The fact that rewards are always distributed at the end of a learning episode in our setup allowed us to accumulate covariances throughout the episode, instead of making use of eligibility traces to keep track of the covariances in the recent past. This, in fact, makes it easier to solve the distal reward problem, since credit can only be assigned to exploration that happened during the same trial that the reward was received.

Our study of RMH learning in neural networks shows that the considered learning rule works for a wide range of conditions (trainability and observation functions) and for very different tasks. In addition, we have interpreted the network configuration of task 2, in which an entire subnetwork was not trainable, as a model for partially embodied computation. An interesting question in this case is whether learning in the trainable part of the neural network was fundamentally necessary in order to recognize the input patterns, or, in contrast, whether the necessary computations were already present in the network dynamics. In this case, as in a traditional reservoir computing setup, the learning only needed to provide a suitable mapping from the internal dynamics to the observation function.

The presented results show that after training, the system state evolves along a fixed number of highly robust trajectories. This phenomenon is not commonly observed in reservoirs without trained feedback weights, indicating that training half of the network at least provides feedback loops in addition to a suitable observation function. However, whether an actual trainable neural network has added value on top of these functionalities is not clear from the current experimental results. A more detailed analysis of the learning outcomes in assemblies of fixed and trainable substrates, as a model for partially embodied computation, is the subject of ongoing work.

Given the promising results we obtained in our simulations and the limited number of assumptions we had to make to obtain successful results, this work paves the way toward more complex control hierarchies for robot motor control, in which each level refines the output of the previous one. In this context too, our results raise some interesting questions, for instance, about the exact role of the very poorly performing kinematic controller in our experiments. In fact, the main goal of the kinematic controller is not to have the optimal performance, but rather to inject energy into the structure. In our previous work, we showed an example with an instantaneous reward function in which we first trained a feedback controller with known target signals using recursive least squares, and then proceeded to learn additional feedback signals using a reward-modulated Hebbian rule (Caluwaerts et al., 2012, section 5.1.3). The reason why an additional energy source is employed in both cases, is that it is hard to consistently learn pure feedback controllers with simple Hebbian-like learning rules. A small change in a feedback weight can cause the system dynamics to fade out, which often results in instability. Therefore, an easy and efficient solution is to use an additional input that consistently pumps energy into the system. In principle, this can be accomplished using a feedback controller as in our previous work, a simple feed forward controller as we use here, or another controller, such as a central pattern generator. More extensive research is needed to determine how much of the workload can be offloaded to the lowest level, partially embodied feedback controller and how this scales to more complex control tasks, e.g., involving more complex robot bodies.

In summary, our main conclusion is that reward-modulated Hebbian plasticity provides a simple, yet effective tool for bridging learning in recurrent neural networks and the exploitation of the own dynamics of compliant robots. This strengthens our belief that both the body and the neural network can be used as computational tools and that they should be combined in a self-organizing way into partially embodied hierarchical controllers.

Author Contributions

JB and KC performed the experiments. All authors contributed to the interpretation and structuring of the results and to writing the paper.

Acknowledgments

We would like to thank Andrea Soltoggio for useful discussions in the preparation of this letter. *Funding*: This research was funded by a Ph.D. fellowship of the Research Foundation – Flanders (FWO) and the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreements No. 248311 (AMARSi), and No. 604102 (Human Brain Project).

References

Bache, K., and Lichman, M. (2013). *UCI Machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Sciences. Available at: http://archive.ics.uci.edu/ml

Caluwaerts, K., Despraz, J., Iscen, A., Sabelhaus, A. P., Bruce, J., Schrauwen, B., et al. (2014). Design and control of compliant tensegrity robots through simulation and hardware validation. *J. R. Soc. Interface* 11, 98. doi:10.1098/rsif.2014.0520

Caluwaerts, K., D'Haene, M., Verstraeten, D., and Schrauwen, B. (2012). Locomotion without a brain: physical reservoir computing in tensegrity structures. *Artif. Life* 19, 35–66. doi:10.1162/ARTL_a_00080

Clopath, C., Longtin, A., and Gerstner, W. (2008). An online Hebbian learning rule that performs independent component analysis. *BMC Neurosci.* 9(Suppl. 1):O13. doi:10.1186/1471-2202-9-S1-O13

Connelly, R., and Back, A. (1998). Mathematics and tensegrity. *Am. Sci.* 86, 142–151. doi:10.1511/1998.2.142

Faisal, A. A., Selen, L. P. J., and Wolpert, D. M. (2008). Noise in the nervous system. *Nat. Rev. Neurosci.* 9, 292–303. doi:10.1038/nrn2258

Fiete, I. R., and Seung, H. S. (2006). Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Phys. Rev. Lett.* 97, 048104. doi:10.1103/PhysRevLett.97.048104

Hansen, N., and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* 9, 159–195. doi:10.1162/106365601750190398

Hebb, D. O. (1949). *The Organization of Behavior*. New York, NY: Wiley.

Hoerzer, G. M., Legenstein, R., and Maass, W. (2012). Computational structures from chaotic neural networks through reward-modulated Hebbian learning. *Cereb. Cortex* 24, 677–690. doi:10.1093/cercor/bhs348

Hyvrinen, A., and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Netw.* 13, 411–430. doi:10.1016/S0893-6080(00)00026-5

Ingber, D. E. (1997). Tensegrity: the architectural basis of cellular mechanotransduction. *Annu. Rev. Physiol.* 59, 575–599. doi:10.1146/annurev.physiol.59.1.575

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cereb. Cortex* 17, 2443–2452. doi:10.1093/cercor/bhl152

Kailath, T., Sayed, A. H., and Hassibi, B. (2000). *Linear Estimation*, Vol. 1. New York: Prentice Hall.

Legenstein, R., Chase, S. M., Schwartz, A. B., and Maass, W. (2010). A reward-modulated Hebbian learning rule can explain experimentally observed network reorganization in a brain control task. *J. Neurosci.* 30, 8400–8410. doi:10.1523/jneurosci.4284-09.2010

Legenstein, R., Pecevski, D., and Maass, W. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Comput. Biol.* 4:e1000180. doi:10.1371/journal.pcbi.1000180

Loewenstein, Y. (2008). Robustness of learning that is based on covariance-driven synaptic plasticity. *PLoS Comput. Biol.* 4:e1000007. doi:10.1371/journal.pcbi.1000007

Loewenstein, Y., and Seung, H. S. (2006). Operant matching is a generic outcome of synaptic plasticity based on the covariance between reward and neural activity. *Proc. Natl. Acad. Sci. U.S.A.* 103, 15224–15229. doi:10.1073/pnas.0505220103

Maass, W. (2014). Noise as a resource for computation and learning in networks of spiking neurons. *Proc. IEEE* 102, 860–880. doi:10.1109/jproc.2014.2310593

Mazzoni, P., Andersen, R. A., and Jordan, M. I. (1991). A more biologically plausible learning rule for neural networks. *Proc. Natl. Acad. Sci. U.S.A.* 88, 4433–4437. doi:10.1073/pnas.88.10.4433

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *J. Math. Biol.* 15, 267–273. doi:10.1007/bf00275687

Pfeifer, R., and Bongard, J. (2007). *How the Body Shapes the Way We Think: A New View of Intelligence*. Cambridge: MIT Press.

Pugh, A. (1976). *An Introduction to Tensegrity*. Los Angeles: University of California Press.

Sanger, T. (1989). Optimal unsupervised learning in a single-layer linear feed-forward neural network. *Neural Netw.* 2, 459–473. doi:10.1016/0893-6080(89)90044-0

Skelton, R. E., and de Oliveira, M. C. (2009). *Tensegrity Systems*. New York: Springer.

Snelson, KD. (1965). *Continuous Tension, Discontinuous Compression Structures*. US Patent 3,169,611. Available at: https://www.google.com/patents/US3169611

Soltani, A., and Wang, X.-J. (2006). A biophysically based neural model of matching law behavior: melioration by stochastic synapses. *J. Neurosci.* 26, 3731–3744. doi:10.1523/JNEUROSCI.5159-05.2006

Soltoggio, A., Lemme, A., Reinhart, F., and Steil, J. J. (2013). Rare neural correlations implement robotic conditioning with delayed rewards and disturbances. *Front. Neurorobot.* 7:6. doi:10.3389/fnbot.2013.00006

Soltoggio, A., and Steil, J. (2013). Solving the distal reward problem with rare correlations. *Neural Comput.* 25, 940–978. doi:10.1162/neco_a_00419

Verstraeten, D., Schrauwen, B., D'Haene, M., and Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Netw.* 20, 391–403. doi:10.1016/j.neunet.2007.04.003

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.

Appendix

A stabilized Reward-Modulated Hebbian Rule

Oja's rule (Oja, 1982) and its extension, the generalized Hebbian algorithm or Sanger's rule (Sanger, 1989), provide a single layer neural network implementation to compute principal components. Contrary to pure Hebbian plasticity, the learning rules are stable, because they force the norm of the weight vectors to unity. Unlike in the unsupervised learning case, reward-modulated rules tend to be stable in practice (i.e., the trained weights remain bounded). However, it can still be useful to control the norm of the weights as this can have practical implications. For example, in a robotics application, this would allow for limiting the required feedback gain and thus the required motor power. From a theoretical point of view, it is also instructive to see how the learning rules employed throughout this paper resemble the now classic rule discovered by Sanger over 20 years ago. In this section, we provide a similar derivation for the RMH learning rule that we studied in this work.

To simplify the notation, we start by defining a number of variables:

$$\boldsymbol{E} = \boldsymbol{Z}^T\boldsymbol{X} \tag{A1}$$

$$r' = r - \bar{r}. \tag{A2}$$

The basic learning rule we studied can now be written in element-wise form as:

$$w_{ij}[n+1] = w_{ij}[n] + \alpha r' e_{ij} \tag{A3}$$

Oja's rule is a first order approximation to the normalization of the weights at every update step:

$$w_{ij}[n+1] = b_i \frac{w_{ij}[n] + \alpha r' e_{ij}}{\| \boldsymbol{w}_i[n] + \alpha r' \boldsymbol{e}_i \|}, \tag{A4}$$

where $\boldsymbol{b}$ contains the desired L_2 norms of the weight vectors.

We now consider the linearization of this rule for small learning rates α. To further simplify the notation, we drop the time index and consider a single output dimension:

$$w_j \approx b \frac{w_j + \alpha r e_j}{\| \boldsymbol{w} + \alpha r \boldsymbol{e} \|}\bigg|_{\alpha=0} + \alpha b \left(\frac{\partial}{\partial \alpha} \frac{w_j + \alpha r e_j}{\| \boldsymbol{w} + \alpha r \boldsymbol{e} \|} \right)\bigg|_{\alpha=0} \tag{A5}$$

A straightforward calculation shows that the part inside the parentheses of the second term can be written as:

$$\frac{\partial}{\partial \alpha} \frac{w_j + \alpha r e_j}{\| \boldsymbol{w} + \alpha r \boldsymbol{e} \|}\bigg|_{\alpha=0, \|\boldsymbol{w}\|=b}$$

$$= \frac{r e_j \boldsymbol{w} + \alpha r \boldsymbol{e} \| - (w_j + \alpha r e_j)(\boldsymbol{w} + \alpha r \boldsymbol{e}) \cdot (r\boldsymbol{e})}{\|\boldsymbol{w} + \alpha r \boldsymbol{e}\|^2}\bigg|_{\alpha=0, \|\boldsymbol{w}\|=b} \tag{A6}$$

$$= \frac{1}{b} r e_j - \left(\frac{r \boldsymbol{w} \cdot \boldsymbol{e}}{b^3} \right) w_j, \tag{A7}$$

assuming $\|\boldsymbol{w}\| = b$ and $\alpha = 0$.

The complete learning rule can therefore be written as:

$$w_j = b \frac{w_j}{b} + \alpha b \left(\frac{1}{b} r e_j - \left(\frac{r \boldsymbol{w} \cdot \boldsymbol{e}}{b^3} \right) w_j \right) \tag{A8}$$

$$= w_j + \alpha r \left(e_i - \frac{w_j}{b^2} \boldsymbol{w} \cdot \boldsymbol{e} \right). \tag{A9}$$

The matrix form of the rule for multiple outputs is given by:

$$\boldsymbol{\Delta W} = \alpha r' (\boldsymbol{E} - \mathrm{diag_v}(\boldsymbol{b})^{-2} \mathrm{diag_v}(\mathrm{diag_m}(\boldsymbol{E}\boldsymbol{W}^T))\boldsymbol{W}), \tag{A10}$$

where the $\mathrm{diag_v}$ operator transforms a vector into a diagonal matrix, and the $\mathrm{diag_m}$ operator transforms a matrix into a vector, containing its diagonal elements. The time complexity to this rule is of the order $\mathcal{O}(mn)$, with m the number of outputs and n the number of inputs. The resulting stabilized learning rule very closely resembles Sanger's rule.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 25 September 2015
doi: 10.3389/fnbot.2015.00010

Distributed recurrent neural forward models with synaptic adaptation and CPG-based control for complex behaviors of walking robots

Sakyasingha Dasgupta [1, 2, 3]*, *Dennis Goldschmidt* [2], *Florentin Wörgötter* [1, 2] *and* *Poramate Manoonpong* [2, 4]

[1] Institute for Physics - Biophysics, George-August-University, Göttingen, Germany, [2] Bernstein Center for Computational Neuroscience, George-August-University, Göttingen, Germany, [3] Laboratory for Neural Computation and Adaptation, Riken Brain Science Institute, Saitama, Japan, [4] CBR, Embodied AI and Neurorobotics Lab, The Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense, Denmark

Walking animals, like stick insects, cockroaches or ants, demonstrate a fascinating range of locomotive abilities and complex behaviors. The locomotive behaviors can consist of a variety of walking patterns along with adaptation that allow the animals to deal with changes in environmental conditions, like uneven terrains, gaps, obstacles etc. Biological study has revealed that such complex behaviors are a result of a combination of biomechanics and neural mechanism thus representing the true nature of embodied interactions. While the biomechanics helps maintain flexibility and sustain a variety of movements, the neural mechanisms generate movements while making appropriate predictions crucial for achieving adaptation. Such predictions or planning ahead can be achieved by way of internal models that are grounded in the overall behavior of the animal. Inspired by these findings, we present here, an artificial bio-inspired walking system which effectively combines biomechanics (in terms of the body and leg structures) with the underlying neural mechanisms. The neural mechanisms consist of (1) central pattern generator based control for generating basic rhythmic patterns and coordinated movements, (2) distributed (at each leg) recurrent neural network based adaptive forward models with efference copies as internal models for sensory predictions and instantaneous state estimations, and (3) searching and elevation control for adapting the movement of an individual leg to deal with different environmental conditions. Using simulations we show that this bio-inspired approach with adaptive internal models allows the walking robot to perform complex locomotive behaviors as observed in insects, including walking on undulated terrains, crossing large gaps, leg damage adaptations, as well as climbing over high obstacles. Furthermore, we demonstrate that the newly developed recurrent network based approach to online forward models outperforms the adaptive neuron forward models, which have hitherto been the state of the art, to model a subset of similar walking behaviors in walking robots.

Keywords: neural control, forward models, recurrent networks, locomotion, adaptive behavior, walking robots, synaptic adaptation

Edited by:
Andrea Soltoggio,
Loughborough University, UK

Reviewed by:
Sebastian Risi,
IT University of Copenhagen, Denmark
Auke Ijspeert,
École Polytechnique Fédérale de Lausanne, Switzerland
Qinggang Meng,
Loughborough University, UK

***Correspondence:**
Sakyasingha Dasgupta,
Riken Brain Science Institute, 2-1 Hirosawa, Wako,
351-0198 Saitama, Japan
sakyasingha.dasgupta@riken.jp

Received: *02 April 2015*
Accepted: *31 August 2015*
Published: *25 September 2015*

Citation:
Dasgupta S, Goldschmidt D, Wörgötter F and Manoonpong P (2015) Distributed recurrent neural forward models with synaptic adaptation and CPG-based control for complex behaviors of walking robots. Front. Neurorobot. 9:10. doi: 10.3389/fnbot.2015.00010

1. Introduction

Walking animals show diverse locomotor skills to deal with a wide range of terrains and environments. These involve intricate motor control mechanisms with internal prediction systems and learning (Huston and Jayaraman, 2011), allowing them to effectively cross gaps (Blaesing and Cruse, 2004), climb over obstacles (Watson et al., 2002), and even walk on uneven terrain (Cruse, 1976; Pearson and Franklin, 1984). These capabilities are realized by a combination of biomechanics of their body and neural mechanisms. The main components of these neural mechanisms include central pattern generators (CPGs), internal forward models, and limb-reflex control systems. The CPGs generate basic rhythmic motor patterns for locomotion, while the reflex control employs direct sensory feedback (Pearson and Franklin, 1984). However, it is argued that biological systems need to be able to predict the sensory consequences of their actions in order to be capable of rapid, robust, and adaptive behavior. As a result, similar to the observations in vertebrate brains (Kawato, 1999), insects can also employ internal forward models as a mechanism to predict their future state (predictive feedbacks) given the current state or sensory context (sensory feedback) and the control signals (efference copies), in order to shape the motor patterns for adaptation (Webb, 2004; Mischiati et al., 2015). Essentially, such a forward model acts as an internal feedback loop, that uses a copy of the motor command, in order to predict the expected sensory input. Comparing this to the actual input, appropriate modulations of this signal or adaptive behaviors can be carried out.

In order to make such accurate predictions of future actions to satisfy changing environmental demands, the internal forward models require some degree of memory of the previous sensory-motor information. However, given that, such motor control happens on a very fast timescale, keeping track of temporal information is integral to such very short-term memory processes. Reservoir-based recurrent neural networks (RNNs) (Maass et al., 2002; Jaeger and Haas, 2004; Sussillo and Abbott, 2009), with their inherent ability to deal with temporal information and fading memory of sensory stimuli, thus provide a suitable platform to model such internal predictive mechanisms. Taking this perspective, here, we utilize a newly developed model of self-adaptive reservoir networks (SARN) (Dasgupta et al., 2013; Dasgupta, 2015), to act as the forward models for sensorimotor prediction. This works in conjunction with other neural mechanisms for motor control and generates complex adaptive locomotion in an artificial walking robotic system. Specifically, by exploiting the adaptive recurrent layer of our model it is possible to achieve complex motor transformations at different walking gaits, which is significantly difficult to achieve by currently existing adaptive forward models employed with walking robots (Dearden and Demiris, 2005; Schröder-Schetelig et al., 2010; Manoonpong et al., 2013).

We present for the first time a distributed forward model architecture using six SARN-based forward models on a hexapod robot, each of which is for sensory prediction and state estimation of an individual robot leg. The outputs of the models are compared with foot contact sensory signals (actual sensory feedback) and the differences between them are used for motor adaptation, in an online manner. This is integrated as part of the neural mechanism framework consisting of (1) single central pattern generator-based control for generating basic rhythmic patterns and coordinated movements, (2) distributed reservoir forward models and (3) searching and elevation action control for adapting the movement of an individual leg based on the forward model predictions, in order to deal with changing environmental conditions. The distributed nature of the SARN-based forward models allows each leg to act independently with its own feedback and adapt to various environmental situations. This has hitherto, been a difficult problem with centralized motor prediction architectures (Dearden and Demiris, 2005; Pfeifer et al., 2007). Although, there have been some influential distributed architectures for locomotion control of insect inspired robots (Beer et al., 1992; Cruse et al., 1998), they are largely reactive without any prediction (forward model) ability at each leg. In this work, our distributed approach to motor prediction can not only significantly decrease computational demands but also enable each leg with inherent memory in order to make predictions based on its history of sensorimotor signals. This naturally lends to flexibility and robustness of the overall locomotive behavior. Furthermore, each SARN forward model can learn to make predictions for multiple different walking gaits, which was also hitherto not possible in the current state of the art adaptive neuron forward model architecture (Manoonpong et al., 2013). Additionally, the ability to deal with sensorimotor noise or missing information (corrupt signals) of motor commands can be crucial under real environmental conditions. In this work, we will show that the long internal memory of recurrent neural networks naturally allow our forward models to be noise robust and deal with such abnormal conditions to produce truly adaptive locomotion. Overall the neural mechanisms framework presented in this paper makes primary contributions toward making better controllers for insect inspired legged robots (Ijspeert, 2014). While at the same time the developed adaptation mechanism could also suggest a possible role in animal motor control, given the biological plausibility of a distributed neural architecture (Beer et al., 1992) and local leg control (Berg et al., 2015).

In the following section we describe the architectural setup of the neural mechanisms used for the design of adaptive locomotion control in a walking robot, along with a description of the simulated hexapod robot AMOS II and the modular robot control environment used as the development platform for our proposed control system. In Section 3, we present the materials and methods used in this study. Specifically, we introduce the setup and implementation of the distributed reservoir-based adaptive forward model, with details of the learning procedure. Section 4 presents experimental results of the learning mechanism and the resulting behaviors of the simulated hexapod AMOS II on different complex locomotion scenarios likes crossing a large gap, walking on uneven (rough) terrains, overcoming obstacles and dealing with leg damage scenarios. The results obtained from the reservoir based forward models are juxtaposed with the previous state of the art adaptive neuron forward models setup. Finally, in Section 5, we discuss our results and provide an outlook of further future directions.

2. Neural Mechanisms for Complex Locomotion

The neural mechanisms (**Figure 1A**) for locomotion control, are designed based on a modular architecture, such that, they comprise of, (i) central pattern generator (CPG)-based control, (ii) reservoir-based adaptive forward models, and (iii) searching and elevation action control. The CPG-based control and the searching and elevation control have been previously discussed in detail in Manoonpong et al. (2013), thus here we will only provide a brief overview of these mechanisms, while the reservoir-based adaptive forward models, which forms the main topic of this work, will be presented in detail in the following section.

The CPG-based control primarily generates a variety of rhythmic patterns and coordinates all leg joints of a simulated hexapod robot AMOSII (**Figure 1B**), thereby, leading to a multitude of different behavioral patterns and insect-like leg movements. The patterns include omnidirectional walking and insect-like gaits (Manoonpong et al., 2013). All these patterns can be set manually, or autonomously driven by exteroceptive sensors, like a camera (Zenker et al., 2013), a laser scanner (Kesper et al., 2013), or range sensors. While the CPG-based control provides versatile autonomous behaviors, the searching and elevation control at each leg uses the accumulated error signals provided by the reservoir-based adaptive forward models in order to adapt the movement of an individual leg of the robot and deal with changes in environmental conditions.

The CPG-based control (see Supplementary Figure 1 for detailed description) itself is designed as a modular neural network that consists mainly of the following four elements:

1. CPG mechanism with neuromodulation for generating different rhythmic signals. Inspired by biological findings, here the CPG circuit is designed as a two-neuron fully connected recurrent network (Pasemann et al., 2003) (Supplementary Figure 1, top left), such that using different external neuromodulatory inputs different walking gaits can be achieved.
2. CPG post-processing units (PCPG) for shaping CPG output signals.

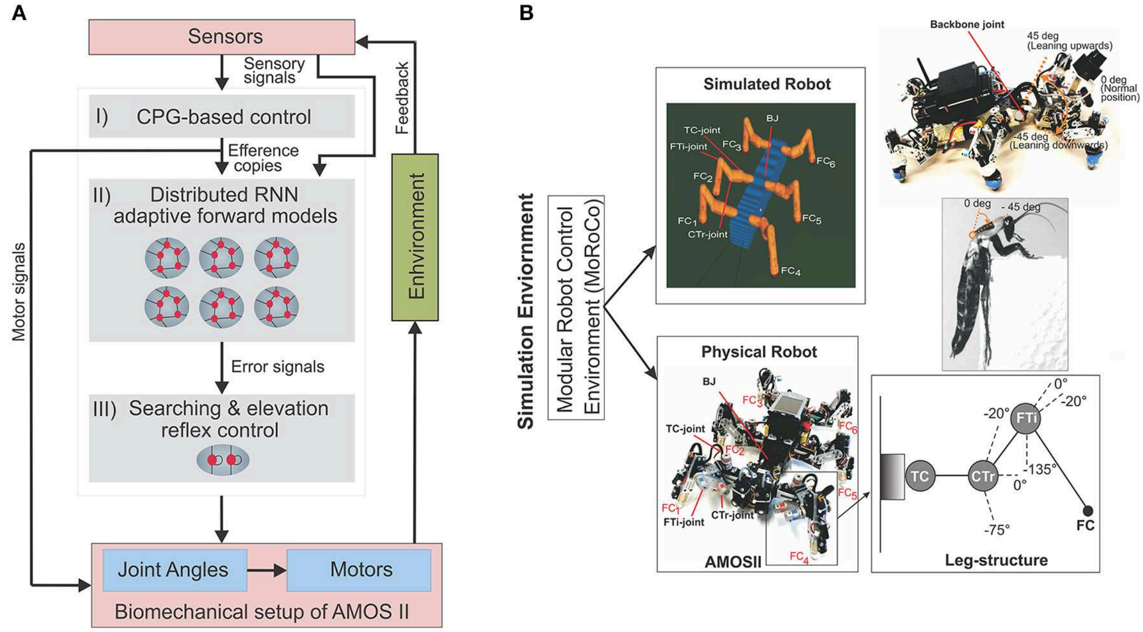


FIGURE 1 | (A) The closed-loop architectural diagram of an artificial bio-inspired walking system consisting of the sensors (i.e., proprioceptive and exteroceptive sensors) that receive environmental inputs and feedback, the neural mechanisms (i, ii, iii) for adaptive locmotion control, and the biomechanical setup of the hexapod robot AMOSII [i.e., six 3-jointed legs, a segmented body structure with one active backbone joint (BJ), actuators, and passive compliant components Manoonpong et al., 2013]. **(B)** Modular Robot Control Environment embedded in the LPZRobots simulation toolkit (Der and Martius, 2012; Hesse et al., 2012). (Top left) The simulation environment provides the main testbed for developing the controller, testing it on the simulated hexapod robot, and finally transferring it to the physical agent. Here we evaluate our model and results primarily on the simulated robot (bottom left), which accurately embodies the characteristics of its physical equivalent, AMOS II robot (bottom left). Here, FC_1, FC_2, FC_3, FC_4, FC_5, and FC_6 are foot contact sensors installed in the robot legs, which are used as the main sensory stimuli compared against the predicted signal from the RNN-based (reservoir) forward models. Each leg (bottom right inset) consists of three joints: the innermost thoraco-coxal (TC-) joint enables forward and backward movements, the middle coxa-trochanteral (CTr-) joint enables elevation and depression of the leg, and the outermost femur-tibia (FTi-) joint enables extension and flexion of the tibia. The morphology of these multi-jointed legs were designed based on a cockroach leg (Zill et al., 2004). (Top right) The front and back parts of the body are connected with a backbone joint (BJ) which primarily allows upwards and downwards tilting of the front body segment (along the horizontal axis). Thus, this is used for climbing and gap crossing purposes. This is also based on a similar joint structure found in the cockroach morphology, allowing it to climb large obstacles. More details on BJ control for climbing can be found in Goldschmidt et al. (2014).

3. Phase switching network (PSN) and velocity regulating networks (VRNs) for walking directional control.
4. Motor neurons with embedded fixed delay lines for transmitting motor commands to all leg joints of AMOS II. These delay lines are utilized to realize the inter-limb coordination, in which they introduce phase differences between the transmitted signals to all leg joints. As a result, the desired walking gait can be achieved.

All neurons of the control network are modeled as discrete-time rate-coded neurons. They are updated with a frequency of approximately 27 Hz (1 time step ≈37 ms). The activity ϕ_i of each neuron in the control network develops according to:

$$\phi_i(t) = \sum_{j=1}^{n} W_{ij}\theta_j(t-1) + \epsilon_i, i = 1, \ldots, n. \tag{1}$$

where, n denotes the number of units, ϵ_i is an internal bias signal or stationary input to each unit i, W_{ij} are the synaptic strength of the connections from neuron j to neuron i. The output θ_i of all neurons of the control network are calculated by using the hyperbolic tangent (tanh) transfer function, i.e., $\theta_i = tanh(\phi_i)$, $\in [-1, 1]$, except for the CPG postprocessing neurons use a step function, the motor output neurons use a piecewise linear transfer function.

The discrete time dynamics of activity states ϕ_i of the two-neuron ($i \in 1, 2$) fully connected CPG circuit, and its output states θ_i follows Equation (1) and a tanh transfer function, respectively. The initial states of the CPG neurons are set to a small positive value, e.g., 0.1. An external excitatory modulatory input MI is introduced to the synaptic connections of the neurons (Supplementary Figure 1, above), in order to modulate the outputs of the CPG. Here different values of MI generates different walking gait patterns (wave, tetrapod, catterpillar, tripod etc.). Although this can be set automatically using sensory inputs (Manoonpong et al., 2013), here we set their values by hand using empirical evaluations. As such, the synaptic weights of the CPG circuit follows:

$$W_{11,22} = d_0, \tag{2}$$
$$W_{12m} = W_{d1} + MI, \tag{3}$$
$$W_{21m} = -(W_{d1} + MI). \tag{4}$$

where, $W_{11,22}$ are fixed synapses with value $d_0 = 1.4$ and $W_{12m,21m}$ are modulated or plastic synapses. Here, W_{d0} and W_{d1} are default synaptic weights selected such that basic periodic signals can be generated. They need to be selected in accordance with the dynamics of the system that generates periodic or quasi-periodic attractors (Pasemann et al., 2003).

The searching and elevation control at each leg, consist of single recurrent neurons that receive the difference (instantaneous error) between the predicted forward model signal and the actual sensory feedback. Due to the recurrent self-connection, this error is accumulated over time. The accumulated error can then be used to either extend specific leg joints in order to get better foothold (searching action) during the stance phase, or elevate further to overcome obstacles during the swing phase (see **Figure 6E** in Section 4.1). Similar to the CPG-based control, all neurons in the searching and elevation control are modeled as discrete-time rate-coded neurons with piece-wise linear activation functions (see Manoonpong et al., 2013, for details), respectively.

3. Materials and Methods

3.1. Reservoir-based Distributed Adaptive Forward Models

We design, six identical adaptive RNN-based forward models ($RF_{1,2,3,\ldots,6}$), one for each leg of the walking robot (**Figure 2A**). These serve the purpose of online sensorimotor prediction as well as state estimation. Specifically, each forward model learns to correctly transform the efference copy of the actual motor signal for each leg joint (i.e., here the CTr-joint motor signal)[1], into an expected or predicted sensory signal. This predicted signal is then compared with the actual incoming sensory feedback signals (i.e., here the foot contact signal—**Figure 2B**, of each leg) and, based on the error accumulated over time, it triggers the appropriate action (searching or elevation) and modulate the locomotive behavior of the robot. Each forward model is based on a random RNN architecture of the self-adaptive reservoir network type (Dasgupta et al., 2013; Dasgupta, 2015). Due to the presence of rich recurrent feedback connections, the dynamic reservoir and intrinsic homeostatic adaptations, the network exhibits a wide repertoire of non-linear activity and long fading memory. This can be primarily exploited for the purpose of specific leg joint-motor signal transformation, act as motor memory and for the prediction of sensorimotor patterns arising in the current context.

3.2. Network Setup

The basic setup of each reservoir forward model can be divided into three layers: input, hidden (or internal), and readout layers (**Figure 2B**). The internal layer consists of a large recurrent neural network driven by time-varying stimuli (CPG motor signals). These driving signals are projected via the input layer. The internal layer is constructed as a random RNN with fixed randomly initialized synaptic connectivity (in this setup we only modify the reservoir-to-readout neuron weights). Using a discrete time version of SARN, with a step size of Δt, the discrete time state dynamics of each reservoir neuron is given by the following equations:

$$x_i(t+1) = \left(1 - \frac{\Delta t}{\tau_i}\right) \; x_i(t) + \frac{\Delta t}{\tau_i}\left(g\sum_{j=1}^{N} W_{i,j}^{rec} r_j(t) + W_{i,1}^{in} u(t) + B_i\right), \quad i = 1, \ldots, N. \tag{5}$$

$$r_i(t) = \tanh(a_i x_i(t) + b_i), \tag{6}$$

$$\mathbf{z}(t) = \left[\mathbf{W}^{out}\right]^T \mathbf{r}(t). \tag{7}$$

[1] We use the CTr-joint motor signal instead of the TC- and FTi-motor signals since this shows clear swing (off the ground) and stance (on the ground) phases which can be qualitatively matched to the actual foot contact signal.

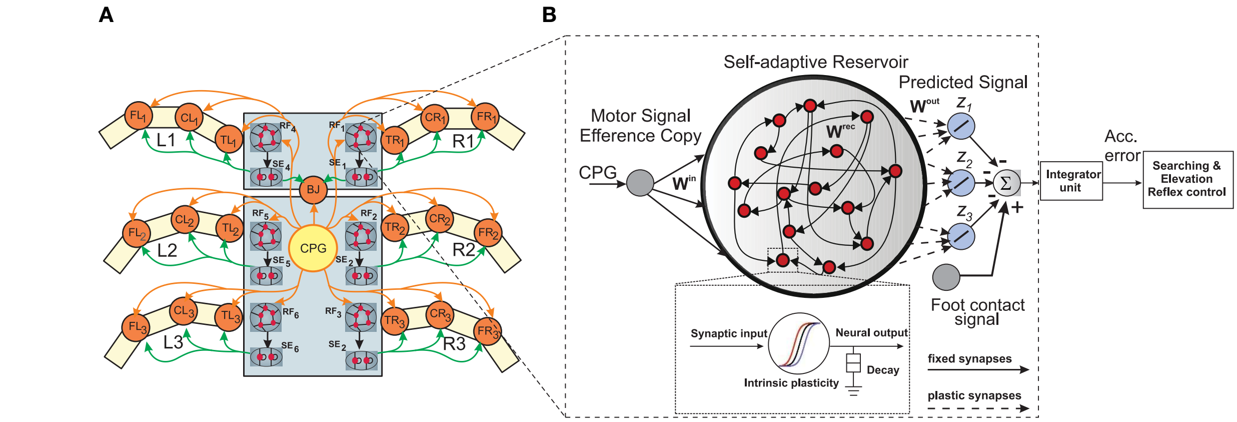


FIGURE 2 | (A) Neural mechanisms implemented on the bio-inspired hexapod robot AMOSII. The yellow circle (*CPG*) represents the neural locomotion control mechanism (see Supplementary Figure 1). The gray circles ($RF_{1,2,3,...,6}$) represent the reservoir-based adaptive forward models. The green circles ($SE_{1,2,3,...,6}$) represent searching and elevation control modules. The orange circles represent leg joints where TR_i, CR_i, FR_i are TC-, CTr- and FTi-joints of the right front leg ($i = 1$), right middle leg ($i = 2$), right hind leg ($i = 3$) and TL_i, CL_i, FL_i are left front leg ($i = 1$), left middle leg ($i = 2$), left hind leg ($i = 3$), respectively. *BJ* is a backbone joint. The orange arrow lines indicate the motor signals which are converted to joint angles for controlling motor positions. The black arrow lines indicate error signals. The green arrow lines indicate signals for adapting joint movements to deal with different circumstances. **(B)** An example of the reservoir-based adaptive forward model. The dashed frame shows a zoomed in view of a single reservoir neuron. In this setup, the input to each of the reservoir network comes from the CTr-joint of the respective leg. The reservoir learns to produce the expected foot contact signal for three different walking gaits (z_1, z_2, z_3). The signals of the output neurons are combined and compared to the actual foot contact sensory signal. The error from the comparison is transmitted to an integrator unit. The unit accumulates the error over time. The accumulated error is finally used to adapt joint movements through searching and elevation control.

The RNN model consists of N neurons, such that the membrane potential at the soma (at time t) of the reservoir neurons, resulting from the incoming excitatory and inhibitory synaptic inputs, is given by a N dimensional vector of neuron state activations. $x(t) = x_1(t), x_2(t), ..., x_N(t)$. The RNN here, does not explicitly model action potentials, but describes neuronal firing rates. Where in, the variable $r_i(t)$ describes the instantaneous firing rate (N dimensional) of the reservoir neurons and is calculated as a non-linear function of the state activation $x_i(t)$ (Equation 5). Each reservoir neuron i, receives inputs from other neurons in the network with firing rates $r_j(t)$ via synaptic connections of strength W_{ij}^{rec} along with incoming stimuli from the input layer via synapses of strength W_{ij}^{in}. Each reservoir neuron is also provided with an auxiliary bias B_i. The parameter g (Sompolinsky et al., 1988; van Vreeswijk and Sompolinsky, 1996) acts as the scaling factor for the recurrent connection weights allowing different dynamic regimes from stable ($g < 1$) to highly irregular chaotic ($g > 1$) (Sussillo and Abbott, 2009), being present in the network.

The input to the reservoir $u(t)$, consists of a single CTr-joint motor signal. This acts as an efference copy of the post-processed CPG motor output. The readout layer consists of three neurons, with their activity being represented by the three-dimensional vector $\mathbf{z}(t)$. Although typically $M < N$ readout neurons can be connected to the reservoir, here we restricted it to three neurons, as each readout here learns the predictive signal for one of the following different walking gaits: wave (z_1), tetrapod (z_2), and caterpillar (z_3) gaits. The wave, tetrapod, and caterpillar gaits are used for climbing over an obstacle, walking on uneven terrain, and crossing a large gap, respectively[2]. Subsequent to the supervised training of the reservoir-to-readout connections $\mathbf{W}^{out}$, each readout neuron basically learns to predict the expected foot contact signal associated with each of these gaits. The decay rate for each reservoir neuron is given by $\frac{1}{\tau_i}$, where τ_i is the individual membrane time constant. The input-to-reservoir connections weights $\mathbf{W}^{in}$ and internal recurrent weights $\mathbf{W}^{rec}$ were drawn randomly from the uniform distribution $[-0.1, 0.1]$ and a Gaussian distribution of zero mean and variance $\frac{g^2}{\sqrt{p_c N}}$, respectively. Where, the parameter p_c controls the probability of connections inside the recurrent layer and is set to be 20%. In order to select the appropriate reservoir size, empirical evaluations were carried out (**Figures 3A,B**), such that we achieved a moderate network size of $N = 30$, for which the minimum prediction error was obtained at the readout layer, irrespective of the walking gait. The recurrent weights were subsequently scaled by the factor of $g = 0.95$ (see **Figure 3**), such that the spontaneous network dynamics is in a stable regime and achieves the best performance of the chosen network size. In accordance with the SARN model, unsupervised intrinsic plasticity (Triesch, 2005) and neuron timescale adaptation (Dasgupta, 2015) were carried out in order to learn the transfer function parameters (a_i and b_i) and the

[2] These three gaits were empirically selected among 19 other possibilities. Previous studies have demonstrated that the wave and tetrapod gaits are the most effective for climbing and walking on uneven terrains, respectively. While in this particular study we observed that the caterpillar gait was the most effective for crossing a gap. However, without any loss of performance, additional walking gaits can be applied easily by adding further readout neurons.

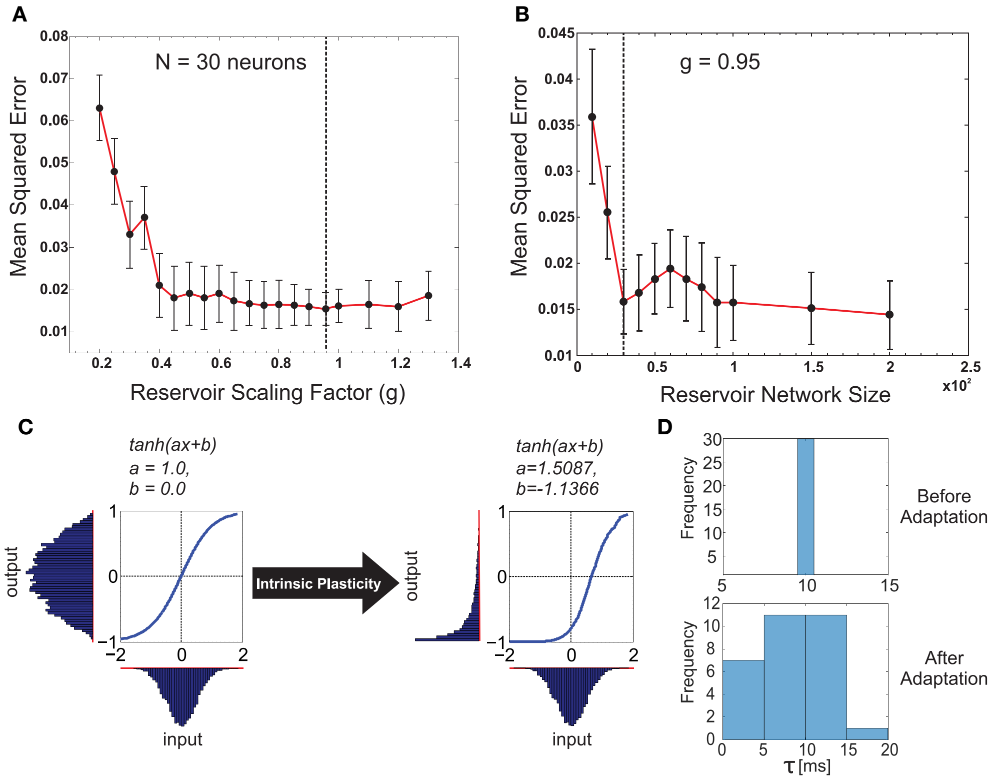


FIGURE 3 | (A) Plot of the change in the mean squared error for the forward model task for one of the front legs (R_1) of the walking robot with respect to the scaling of the recurrent layer synaptic weights W^{rec} with different g-values. As observed, very small values in g have a negative impact on performance compared with values closer to one being better. Interestingly, the performance did not change significantly for $g > 1.0$ (chaotic domain). This is mainly due to homeostasis introduced by intrinsic plasticity in the network. The optimal value of $g = 0.95$ selected for our experiments is indicated with a dashed line. **(B)** Plot of the change in mean squared error with respect to different reservoir sizes (N). g was fixed at the optimal value. Although increasing the reservoir size in general tends to increase performance, a smaller size of $N = 30$ gave the same level of performance as $N = 100$. Accordingly keeping in mind the trade off between network size and learning performance, we set the forward model reservoir size to 30 neurons. Results were averaged over 10 trials with different parameter initializations on the forward model task for a single leg and a fixed walking gait. **(C)** Example of the intrinsic plasticity to adjust the reservoir neuron non-linearity parameters a and b. Initially the the reservoir neuron fires with an output distribution of Gaussian shape matching that of the input distribution. However, after adjustment using intrinsic plasticity mechanism (Dasgupta et al., 2013) the reservoir neuron adapts the parameters a and b, such that, now for the same Gaussian input distribution the output distribution follow a maximal entropy Exponential-like distribution. **(D)** Distribution of the reservoir forward model individual neuron time constants before and after adaptation.

reservoir time constant parameters τ_i for each individual neuron (**Figures 3C,D**).

3.3. Readout Weight Adaptation

Here we used a modified version of the original recursive least squares (RLS) algorithm (Simon, 2002; Jaeger and Haas, 2004) based on the FORCE learning formulation (Sussillo and Abbott, 2009), in order to learn the reservoir-to-readout connection weights $\mathbf{W}^{out}$ at each time step, while the CPG input $u(t)$ is being fed into the reservoir. The readout weights $\mathbf{W}^{out}$ are calculated such that the overall error at the readout neurons is minimized; thereby the network can learn to accurately transform the CTr-motor signal to the expected foot contact signal, for each walking gait. The instantaneous error signal ($e(t)$) at the readout layer, can be calculated as the difference between the reservoir predicted output ($z(t)$) and the desired output, $d(t)$ (i.e., here the expected foot contact signal). Based on Equation (7), this can be formulated as:

$$e(t) = \sum_{j=1}^{3} W_j^{out}(t-1)r_j(t) - d(t). \tag{8}$$

Using the RLS algorithm, and minimizing this error, the readout weights W_j^{out} update can be defined by,

$$W_i^{out} = W_i^{out}(t-1) - e(t)\sum_j P_{ij}(t)r_j(t). \tag{9}$$

Where, $\mathbf{P}$ is a $N \times N$ square matrix proportional to the inverse of the correlation matrix of the reservoir neuron firing rate vector $\mathbf{r}$. $\mathbf{P}$ is initialized using the identity matrix $\mathbf{I}$ and a small constant parameter δ_c as, $\mathbf{P}(0) = \frac{\mathbf{I}}{\delta_c}$. $\mathbf{P}$, here, acts as the adaptive learning rate for updating the readout weights with weight modifications automatically slowing down as $\mathbf{P}$ decreases with time. This allows the learning to occur stably and eventually converge to a solution. $\mathbf{P}$ is updated as each time point as,

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \left(\frac{\mathbf{P}(t-1)\mathbf{r}(t)\mathbf{r}^T(t)\mathbf{P}(t-1)}{1 + \mathbf{r}^T(t)\mathbf{P}(t-1)\mathbf{r}(t)} \right). \tag{10}$$

The reservoir-to-readout neuron weights were initialized to zero at start. Details of all the fixed parameters and initial settings for the reservoir based forward model networks are summarized in Supplementary Table 1.

4. Results

4.1. Learning the Reservoir Forward Model (Motor Prediction)

The entire learning and testing procedure of the SARN-based forward models can be divided into three stages, namely:

1. Pre-training: This stage is used primarily for gathering preliminary sensorimotor data in order to adapt the SARN individual neuron parameters. Here, no reservoir-to-readout weight adaptations occur. This stage can be further divided into,
   - Sequential learning: The robot walks under normal conditions, while sequentially transitioning from one walking gait to another (fixed duration of time). The process is stopped after all the gaits are completed.
   - Offline SARN adaptation: The sensorimotor data collected from the above process is used to adapt the reservoir neuron non-linearity and time constant parameters (Dasgupta et al., 2013).
2. Online training: The same procedure of sequential learning is carried out, however now with ongoing adaptations of the reservoir-readout neuron connection weights based on the RLS algorithm (Equation 9).
3. Testing: After only a single online training cycle the learned forward models are tested on the different experimental conditions.

We now provide a more in depth explanation of these different learning stages.

4.1.1. Pre-training (Without Weight Adaptation)

In order to train the six forward models ($RF_1 to RF_6$) in an online manner, one for each leg, we let the simulated robot AMOSII walk under normal conditions (i.e., walking on a flat terrain with the three different gaits). Initially, we let the robot walk with a certain walking pattern, and then every 2500 time steps (here one time step is equivalent to 37 ms, therefore 2500 time steps is equal to 92.5 s), the gait pattern was sequentially altered (this occurs by changing the modulatory input to the CPG—see Supplementary Figure 1). As a result, the robot sequentially transitions from wave gait, to tetrapod gait, to caterpillar gait repeatedly (here these gaits were empirically selected as the most efficient for the different tasks, however multiple different such gaits can be learned by a single forward model. For an example with commonly used tripod gait, see Supplementary Figure 2). Using this procedure, we let the robot walk for three complete cycles (22,500 time steps) and collected the corresponding CTr-motor signal and foot contact sensor readings for all legs. Intrinsic plasticity and neuron time constant adaptations (Dasgupta et al., 2013; Dasgupta, 2015), were then carried out using 20 epochs of 1000 time steps overlapping time windows. After this pre-training phase, all the reservoir neuron non-linearity parameters and individual time constants (τ_i) were fixed (see **Figure 3D** for the distribution of neuronal time constants before and after training).

4.1.2. Online Training (With Weight Adaptation)

Subsequent to the pre-training phase, normal training of the reservoir-to-readout weights $\mathbf{W}^{out}$ was carried out using the online RLS learning algorithm with the same process of making the robot walk on a flat, regular terrain and sequential switching between the three gait patterns every 2500 time steps. As such, at any given point in time only one of the readout neurons (specific to the walking gait) are active. In this manner, synaptic weights projecting from reservoir to the first readout neuron (z_1) corresponding to the foot contact signal prediction for the wave gait, and synaptic weights projecting to the second (z_2) and third (z_3) readout neurons corresponding to the foot contact signal prediction of the tetrapod and caterpillar gaits, are learned, respectively. Within this experimental setup, as observed from **Figures 4A–C** the readout weights corresponding to each gait converges very quickly, in less than the trial period of 2500 time steps[3]. As a result, every time the CTr-motor signal changes due to walking gait transformations, the RF associated with each leg learns to predict the expected foot contact signal robustly. The training process was carried out only once under normal walking conditions. This was subsequently used as the baseline in order to compare with the actual foot contact signals (sensory feedback) while walking under the situations of crossing a gap, climbing, and negotiating uneven terrains.

Figure 5 shows an example of the forward model prediction (training) during the three different walking gaits, for the right front leg of AMOSII (R_1). Visual inspection clearly demonstrates that according to the corresponding efference copy of CTr-motor signal at a particular gait, the expected foot contact (FC) signal is precisely predicted at each time point. Similarly, the foot contact signals for the other legs are also predicted online, given the current context of CTr-signal (not shown). Note that the FC signals of the other legs normally show slightly different periodic patterns. Furthermore, there exists considerable lag between the expected stance phase according to the motor signal and that observed from the FC signal (difference between dotted green lines in **Figure 5**). Due to the internal memory of the incoming motor signal in the reservoir, we see that the output neurons can adapt to these time lags efficiently, even when the frequency of the signal increases with a change in walking gaits. Furthermore, the reservoir-based forward models enable the robust generation of the predicted FC signal, even in the presence of high noise corruption or missing information in the incoming CTr-joint motor signal (**Figures 5J,K**). Due to the fact that the CTr-motor signals are obtained after appropriate

[3] Due to intrinsic noise and nature of the reservoir-to-readout synaptic adaptation, the weights still show minute fluctuations after successful learning; therefore here convergence applies that the norm of the readout weights $|W^{out}|$ remains constant with a small finite value (Sussillo and Abbott, 2009).

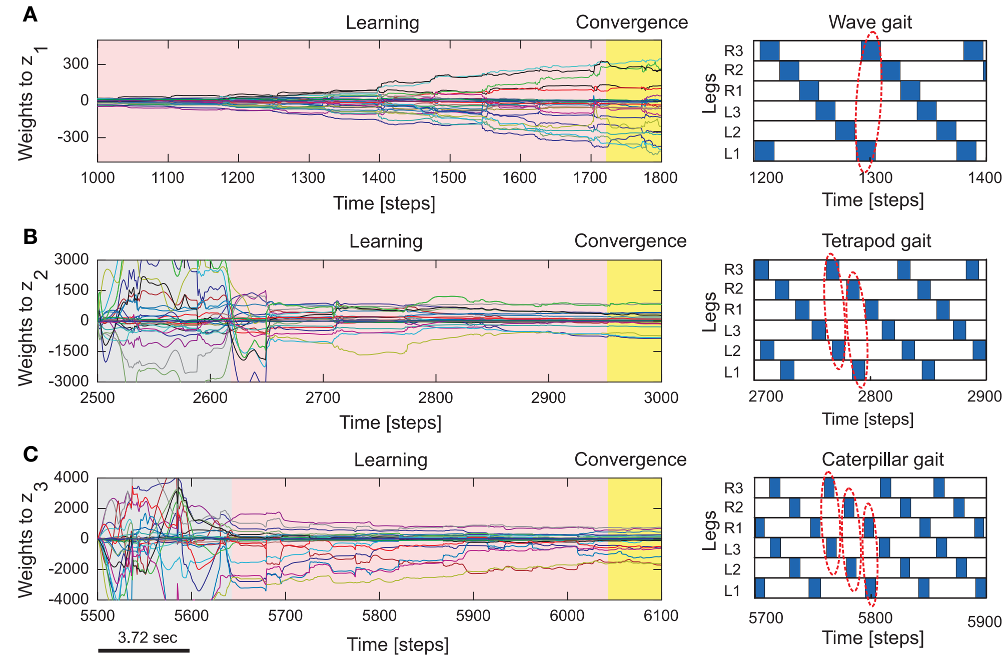


FIGURE 4 | Reservoir-to-readout weight adaptation during online learning. (A) Changes of 30 weights projecting to the first readout neuron (z_1) of the forward model of the right front leg (R_1) while walking with a wave gait. During this period, weights projecting to the second (z_2) and third (z_3) output neurons remain unchanged (i.e., they are zero). **(B)** Changes of the weights to z_2 while walking with a tetrapod gait. During this period, the weights to z_3 still remain unchanged and the weights to z_1 converge to around zero. **(C)** Changes of the weights to z_3 while walking with a caterpillar gait. During this period, the weights to z_1 and z_2 converge to around zero. At the end of each gait, all weights are stored such that they will be used for locomotion in different environments. The gray areas represent transition phases from one gait to another gait and the yellow areas represent convergence. The gait diagrams are shown on the right. They are observed from the motor signals of the CTr-joints (**Figure 5**). White areas indicate ground contact or stance phase and blue areas refer to no ground contact during swing phase. As frequency increases, some legs step in pairs (dashed enclosures). Here convergence implies no significant change in the vector norm of the readout weights.

post-processing of original CPG signals and passage through the motor neurons coupled with different time delays. Such signal corruption can occur at various levels. Therefore, the ability of the forward model to deal with such abrupt noise in the motor signals in a robust manner is crucial to the adaptive mechanisms. Furthermore, such signal corruptions can also occur, due to entrainment mechanisms applied for the automatic tuning or adaptation of CPG outputs (Nachstedt et al., 2013). Such online adaptation for sudden motor signal variations, was not possible in the previous state of the art adaptive neuron forward models (Manoonpong et al., 2013). This model inherently lacked the ability to deal with variations in the temporal properties of the signal. As such, a simple square wave matching the timing of the motor signal efference copy was used, providing a limited range of behavior, as well as being biologically implausible. However, here our reservoir-based model can accurately estimate the spatiotemporal properties of the signal and robustly learn the exact shape, as well as the timing of the actual FC signals.

4.2. Simulated Complex Environments

In order to assess the ability of the reservoir-based forward models to generate adaptive complex locomotive behaviors in a neural closed-loop control system (see **Figure 1**), we conducted simulation experiments under different situations including crossing a gap, walking on uneven terrain and climbing over high obstacles (similar to the behaviors observed in real insects). In all cases, we used the same training procedure for the forward models by allowing the robot to walk under normal conditions on a flat even terrain.

During testing of the learned behavior, while AMOSII walks under different environmental conditions and a specific gait, the output of each trained forward model (i.e., the predicted FC signal, **Figure 6A**) is used to compare it to the actual incoming FC signal of the leg (**Figure 6B**). The difference (instantaneous error signal Δ) between them determines the walking state where a positive value ($+\Delta$) indicates losing ground contact during the stance phase and a negative value ($-\Delta$) indicates stepping on or hitting obstacles during the swing phase.

$$\Delta_i(t) = RF_i(t) - FC_i(t). \tag{11}$$

where $i \in \{1, 2, ..., 6\}$ represents each leg of the robot.

Thus, we use the positive value for searching control (**Figure 6D**, above). This is then accumulated through a single recurrent neuron S with a linear transfer function and is always reset to 0.0 at the beginning of swing phase. Similarly, the negative value is used for elevation control (**Figure 6D**, below). The value is also accumulated through a recurrent neuron E with a linear transfer function. These accumulated errors (**Figure 6C**) thus allow the robot leg to be either elevated (on hitting an

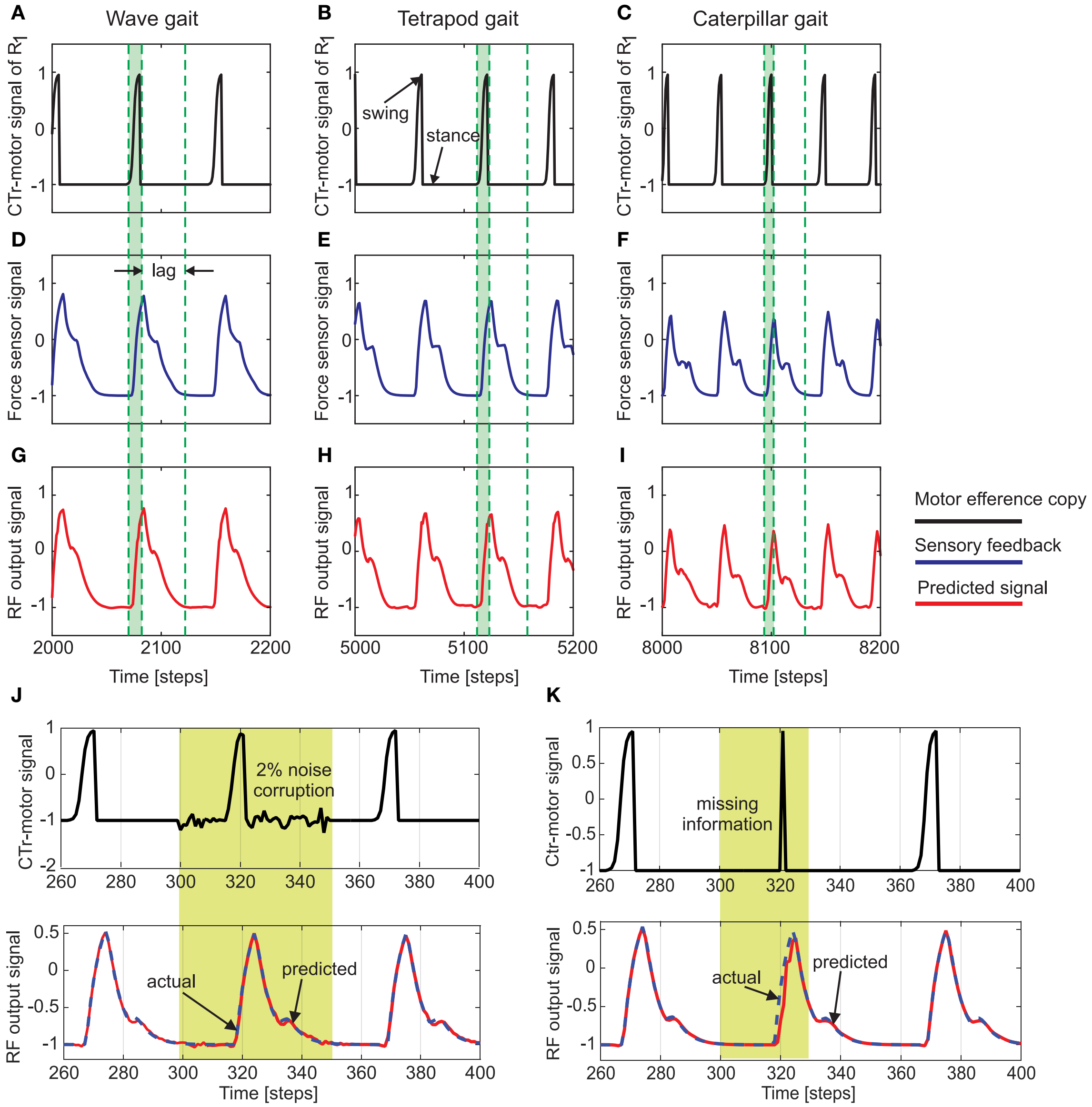


FIGURE 5 | (A–C) The CTr-joint motor signal of the right front leg (R_1) for wave, tetrapod, and caterpillar gaits, respectively. This motor signal provides the efference copy or the input to the reservoir forward models. **(D–F)** The actual foot contact signal (force sensor signal under normal walking conditions) used as the target signal of the reservoir models. **(G–I)** The predicted foot contact signal or the final learned output of the forward model for each walking gait (*RF* output signal). The green shaded region indicates the time interval between swing and stance phase for the CTr motor signal at the three walking gaits. As observed the actual foot contact signal is considerably lagged in time compared to the motor signal. Effectively, this lag decreases with an increase in the gait frequency. The single RF adaptively accounts for these different delay times in order to accurately predict the expected foot contact signal. **(J)** above—CTr-joint motor signal demonstrated for a single leg, with 2% Gaussian noise injected between 300 and 350 time steps (yellow shaded region), below—Despite the noise corruption of the motor signal, the reservoir forward model is able to generate the correct predicted FC signal (blue dotted—target FC signal, red solid—predicted signal). **(K)** above—The CTr-joint motor signal corrupted with missing information between 280 and 320 time steps. As a result, the motor signal shows a narrow spike between 310 and 330 time steps (yellow shaded region), below—Reservoir forward model predicted signal (red) as compared to the desired FC signal (dotted blue). Although the CTr motor signal was transiently missing, the reservoir is able to generate the desired FC signal considerably well, while at the same time maintaining the correct temporal sequence of the signals.

obstacle) or searching for a foothold during the swing and stance phases, respectively (see Manoonpong et al., 2013, for more details of the searching and elevation control). As depicted in **Figures 6A,B**, while walking on a rough terrain (in this case with tetrapod walking gait), the currently recorded sensory feedback or foot contact sensor reading differs considerably from the reservoir predicted signal. As a result, there is a high accumulation of error between each swing or stance phase (**Figure 6C**). It should be noted that the initial (≈50 time steps) abruptly high amplitude signal observed in the reservoir forward

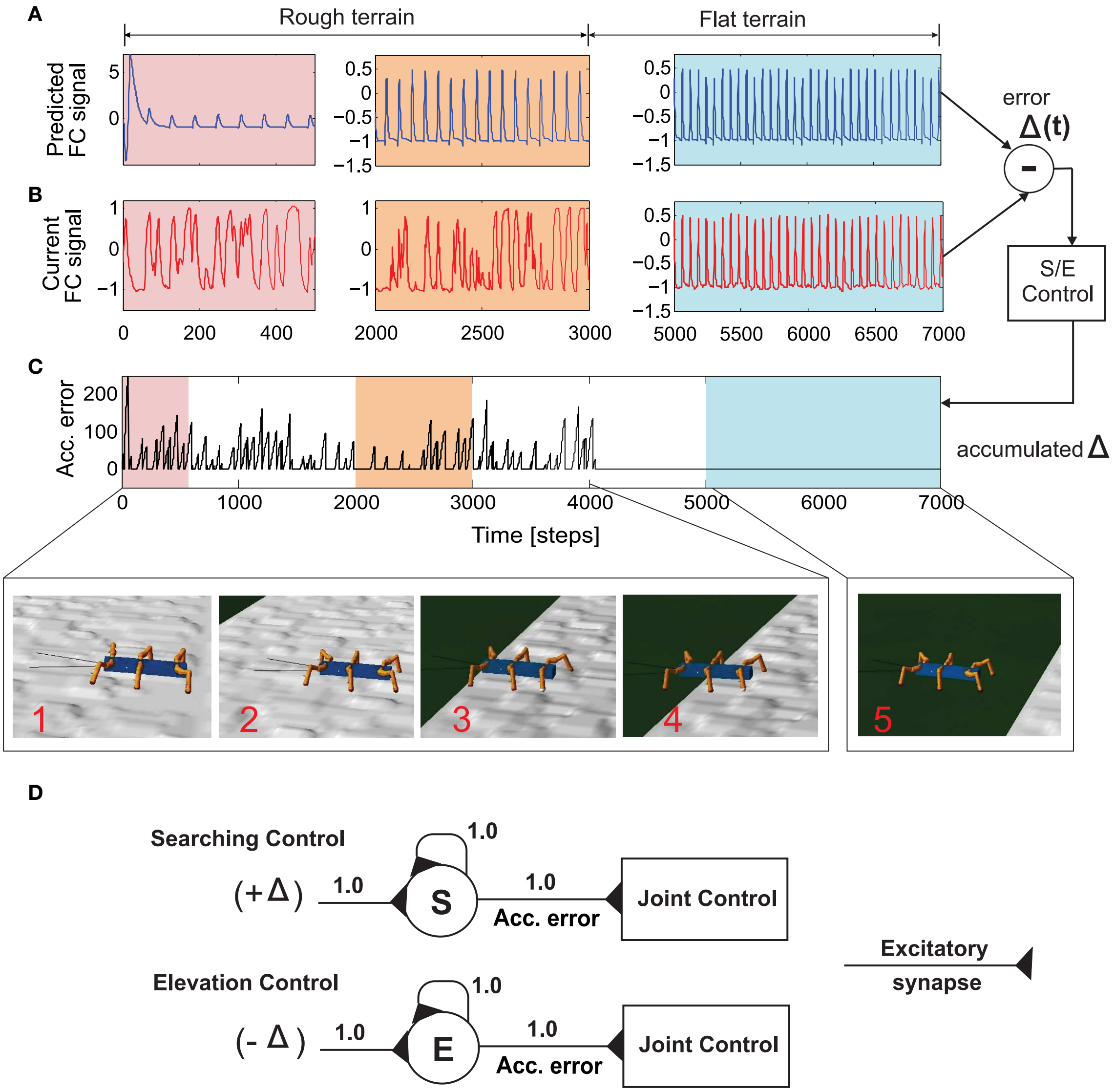


FIGURE 6 | Successfully navigating rough terrain with reservoir forward model. (A) The reservoir forward model predicted, expected foot contact signal. After a small initial transient the reservoir output quickly converges to the expect signal for normal walking condition. **(B)** The actual sensory feedback (foot contact signal) while walking on the rough surface **(C)** Accumulated error calculated from the instantaneous error (Δ(*t*)) after passing through the recurrent neuron in the searching and elevation control. **(D)** The searching and elevation action control system consisting of individual recurrent neurons as signal accumulators. After 4000 time steps, the robot successfully overcomes the rough terrain and continuous walking on a flat surface. As a result, there is zero accumulated error since the predicted foot contact signal almost exactly matches the actual signal. See the experiment Supplementary Video 3.

model prediction, is caused due to the transient recovery time needed by reservoir readout neurons to settle to the exact learned patterns. This is overcome within the next few time steps and RF predicted FC signal continues to occur in a robust manner. The accumulated error causes the corresponding leg action control mechanism to kick in and the robot successfully navigates out of the rough terrain (after ≈4000 time steps). Once the robot moves into the flat terrain, the reservoir predicted foot contact signal matches almost perfectly with the actual sensory feedback. As a result, the accumulated error becomes zero and normal walking without any additional searching or elevation control mechanisms, can continue. In essence based on the reservoir forward models, while traversing from the uneven terrain (**Figure 6**, inset 1–4) to the flat terrain (**Figure 6**, inset 5), the robot can adapt its legs individually to deal with the change of terrain. That is, it depressed its leg and extended its tibia to search for a foothold when loosing ground contact during the stance phase. Losing ground contact information is detected by a significant change of the accumulated errors (**Figure 6C**). In case of both walking on uneven terrain and climbing, this accumulated error causes shifting of the CTr- and FTi-joints causing the respective leg to search for a foothold. However, in the specific case of crossing a gap (**Figure 7**), we use the accumulated error in order to control tilting of the backbone joint (BJ) and shifting of the TC- and FTi-joints such that the front legs can be extended forward continuously till the robot can find a foothold. In addition to this leg joint control, reactive backbone joint control using the additional ultrasonic sensors in front of the robot can also be used to learn to lean up the BJ for climbing over obstacles (this has been previously successfully applied using classical conditioning based learning in Goldschmidt et al. (2014) and as such not discussed here).

We now take the example of the more complex, multiple gap crossing experiment in order to look in detail at the learning outcome of the forward models. This experiment was divided into two components, consisting of one larger gap (15cm length) and another relatively shorter gap of 11 cm length. The two gaps were separated by considerable distance where the robot was allowed to walk on a regular flat terrain. In order to learn to cross a gap, we let AMOS II walk with a caterpillar gait (see **Figure 4C**, right), such that each left and right pair of legs moves simultaneously. Empirically this is observed to be the most suited gait for overcoming large gaps, as well as supported by experimental observations in stick insects (Blaesing and Cruse, 2004). As shown in **Figure 7**(1), at the beginning AMOS II walked forward straight toward the initial gap. In this period, as it walks on the flat surface of the platform, it performed regular movements similar to the training period under normal walking conditions (training on a flat regular surface). Eventually, it encounters a 15 cm wide gap (≈44% of body length—the maximum cross-able distance). In this situation, during the subsequent stance phase the front legs of the robot loose ground contact (**Figures 7D,E**). As a result, the foot contact sensors from the front legs do not record any value. However, the reservoir forward model still predicts the expected foot contact signal, causing a positive instantaneous error (Equation 11). This leads to a gradual ramping of the accumulated error signal between each stance phase and swing phase, for the front legs (**Figure 7A**).

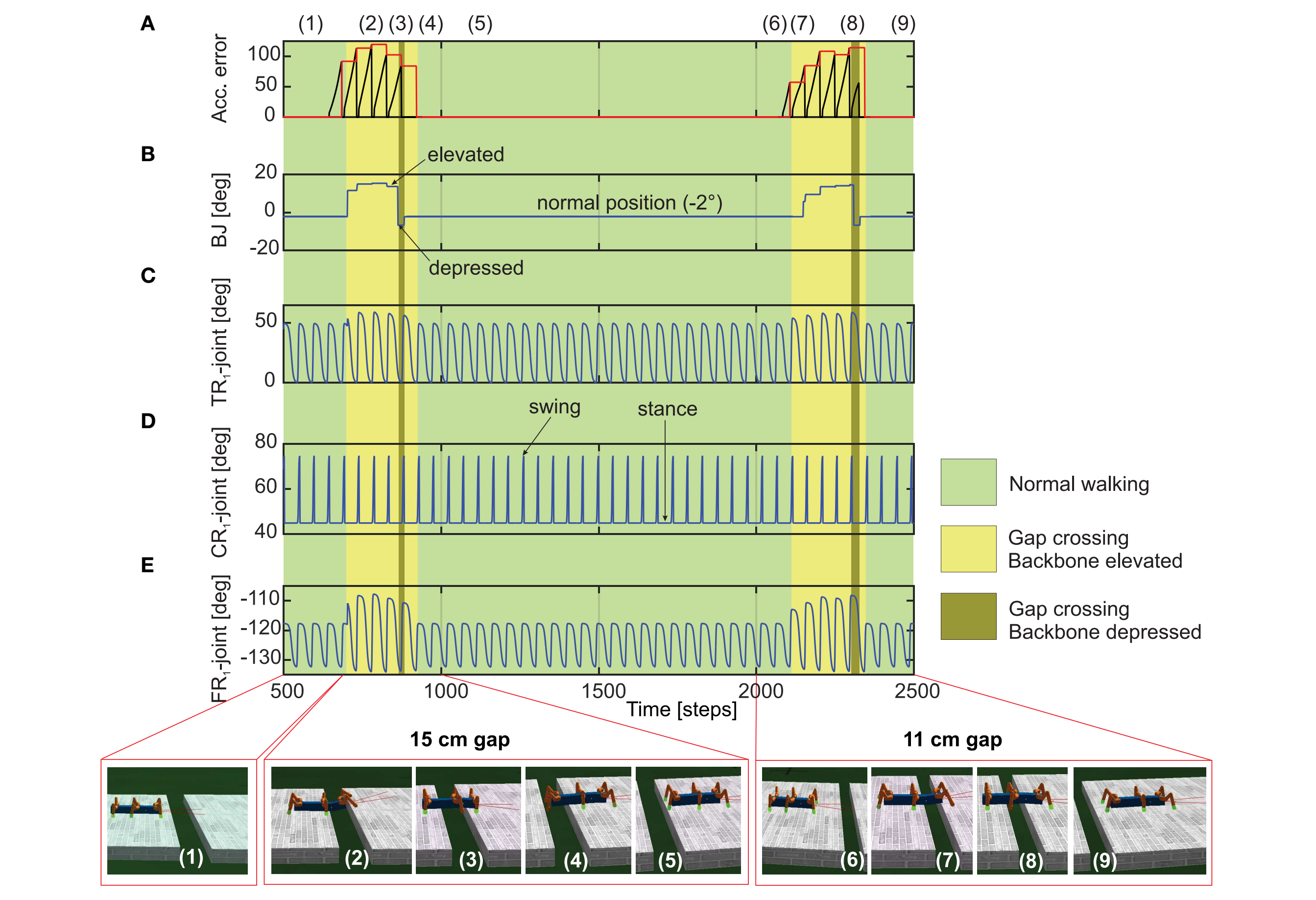


FIGURE 7 | Real-time data of walking and crossing multiple gaps using the forward model predictions. (A) The accumulated error (black line) and the maximum accumulated error value at the end of each stance phase (red line) of the right front leg (R_1). The accumulated error is reset to zero every swing phase. **(B)** The backbone joint (BJ) angle during walking and gap crossing. The BJ stays at the normal position (−2°) during normal walking. On encountering a gap (15cm), it leans upwards in a step like fashion and then finally bent downwards in order to cross the gap. This procedure is repeated for the second gap (11cm), however with different degree of elevations. **(C–E)** The TC-, CTr-, and FTi-joint angles of right front leg R_1 during normal walking and gap crossing. The joint adaptation was controlled by the maximum accumulated error value of the previous step (red line). Below pictures show snap shots of the locomotion of AMOS II during the experiment. Note that one time step is ≈0.037 s. For further details interested readers are recommended to see the experiment Supplementary Videos 1, 2.

Please note that here the slope of the accumulated error signal was empirically adjusted. Too small or too large values for the slope of the ramp may cause inadequate or large extensions of the leg.

In order to activate the BJ and adapt the leg movements due to the difference between the reservoir predicted FC signal and the actual sensory feedback of the FC sensors (error signals), we used the maximum accumulated error value of the previous step (**Figure 7A**, red line) and control the BJ and leg movements in the subsequent step. In this manner, the BJ started to lean upwards incrementally (step like manner) at around 680–850 time steps [**Figure 7**(2)]. Simultaneously, the TC- and FTi-joint movements of the left and right front legs were also adapted accordingly in order to carry out elevation action (this is reflected in the higher amplitude of these two signals in this time period). Due to a predefined time-out period for tilting upwards, at around 850 time steps [**Figure 7**(3)], the backbone joint automatically moved downwards recording a negative value. Consequently, the front legs touch the ground of the second platform at the middle of the stance phase; thereby, causing the accumulated error signals to decrease. Due to another time-out period for tilting downwards at around 900 time steps [**Figure 7**(4)], the BJ automatically moved to the normal position (−2°). Since now the situation is similar to walking on flat terrain, the RF predicted foot contact signal matches the one recorded by the foot sensors, with accumulated error dropping to zero. Thereafter, the TC- and FTi-joints perform regular movements. Subsequently left and right hind legs loose the ground contact, and AMOSII continues to walk forward. Here the movements of the TC- and FTi-joints were slightly adapted allowing AMOS II to successfully cross the gap and continue walking on the second platform [**Figure 7**(5)]. As the terrain now resembles a regular flat surface (similar to the original training terrain) AMOSII two continues to walk forward in normal manner with no accumulated errors being present. However, the same procedure is repeated once again, when AMOSII re-encounters the second gap at around 2100 time steps. However, in this case, since the gap length is much smaller, the elevation in the BJ occurs with an initial increment of smaller amplitude [**Figure 7**(2)] as compared to the previous case. Thereafter, a similar process is followed and AMOSII can once again successfully overcome this gap and continue walking on the other end of the platform [**Figure 7**(9)]. This clearly demonstrates the adaptive yet robust performance of the forward model based predictions in order to successively cross gaps of different length.

Figure 8 shows that the reservoir forward model in combination with the neural locomotion control mechanisms, not only successfully generates gap crossing behavior of AMOS II and learns to walk on uneven terrain, but also allows it to climb over single and multiple obstacles (e.g., up a fleet of stairs). In all these cases, we directly used the accumulated errors for movement adaptation via the searching and elevation control mechanisms. For climbing, the reactive backbone joint control was also applied to the system (see Goldschmidt et al., 2014, for more details) and a slow wave gait walking pattern (see **Figure 4A**, right) was used.

Experimentally the wave gait was found to be the most effective for climbing, which allows AMOSII to overcome the highest climbable obstacle (i.e., 15 cm height which equals ≈86% of its leg length) and to surmount a fleet of stairs. For walking on uneven terrain, a tetrapod gait (see **Figure 4B**, right) was used without the backbone joint control. This is the most effective gait for walking on uneven terrain (see also Manoonpong et al., 2013). Recall that in all experiments the forward models basically generate the expected foot contact signals (i.e., sensory prediction), which are compared to the actual incoming ones. Errors between the expected and actual signals during locomotion serve as state estimation and are used to adapt the joint movements accordingly. It is important to note that, the best gait for each specific scenario was experimentally determined and fixed. However, this could be easily extended with learning mechanisms (see Steingrube et al., 2010) to switch to the desired gait when the respective behavioral

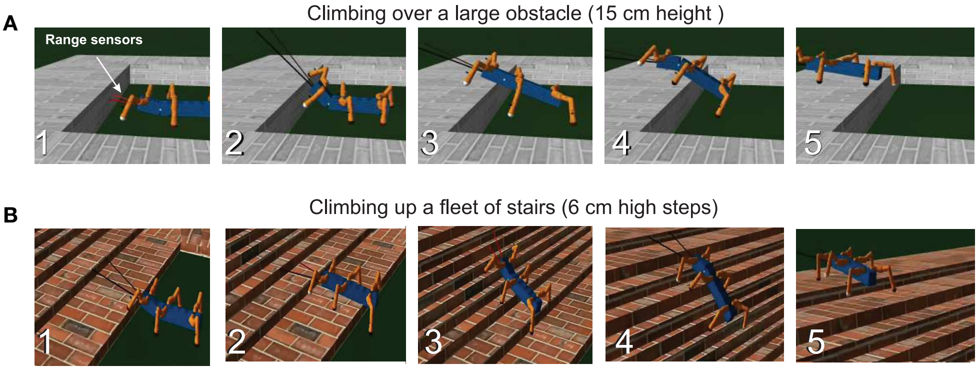

A
Climbing over a large obstacle (15 cm height)
Range sensors
1
2
3
4
5
B
Climbing up a fleet of stairs (6 cm high steps)
1
2
3
4
5

FIGURE 8 | Snapshots showing the learned behavior during climbing over a high obstacle and climbing up a fleet of stairs. (A) AMOSII walked with the wave gait and approached a 15 cm high obstacle (1). It detected the obstacle using its range sensors installed at its front part. The low-pass filtered range sensory signals control the BJ to tilt upwards (2) and then back to its normal position (3). Due to the missing foot contact of the front legs, the BJ moved downwards to ensure stability (4). During climbing, middle and hind legs lowered downwards due to the occurrence of the accumulated errors, showing leg extension, to support the body. Finally, it successfully surmounted the high obstacle (5). For further details see the Supplementary Video 4 **(B)** AMOSII climbed up a fleet of stairs (1–5) using the wave gait as well as the reactive BJ control. The climbing behavior is also similar to the one described in the case **(A)**. For further details see Supplementary Video 5.

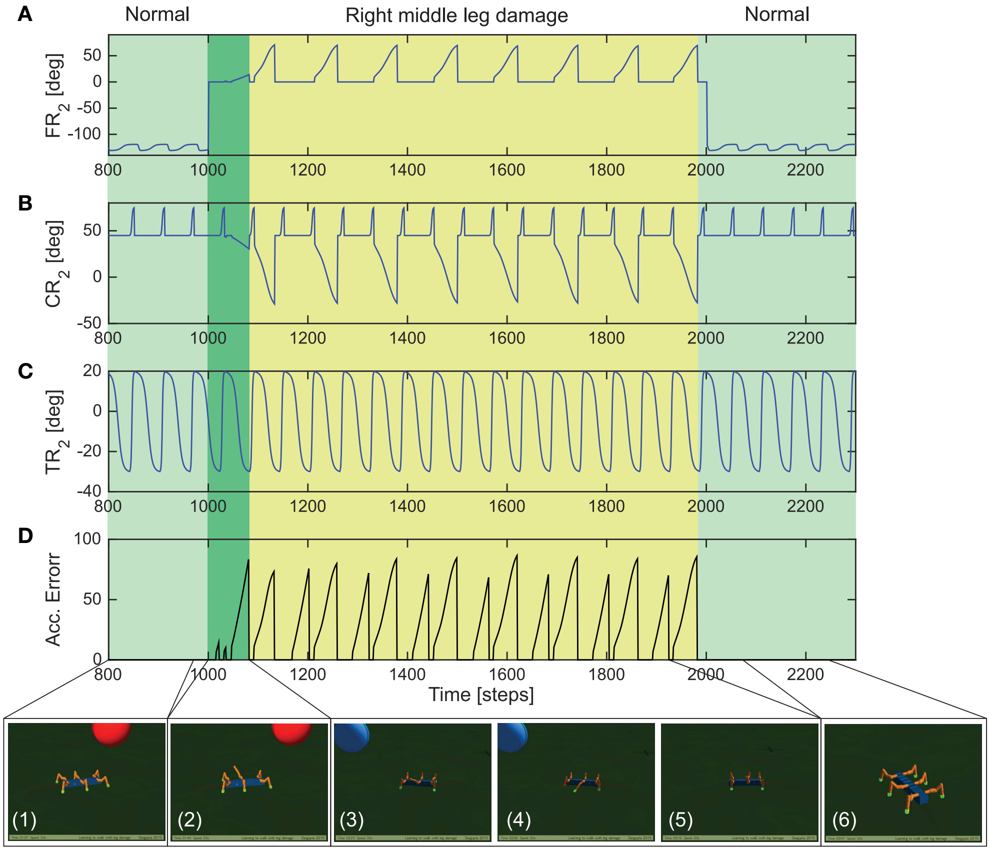


FIGURE 9 | Real-time data for adaptive locomotion to overcome leg damage. (A) The FT-i joint angles of the right middle leg R_2. **(B)** The CT-i joint angles of the right middle leg R_2. **(C)** The TC-i joint angles of the right middle leg. **(D)** Accumulated error signal at the end of each stand phase. It is reset to zero at every swing phase. Below pictures show the locomotion of AMOSII during the experiment (temporal spacing of the panels are not exact). Please see the Supplementary Video 6 for closer look at the exact adaptive behavior.

scenarios are encountered, without any additional influence on the performance of the reservoir forward models.

Adaptations in both biological and robotic systems, not only requires the ability to deal with different environmental conditions for complex locomotion (as demonstrated with the gap crossing, climbing and uneven terrain navigation examples) but can also require the ability to adapt to sudden or abrupt changes in body properties, like growth or lesions (e.g., damage to robot joint motors or connections being disengaged) (Cully et al., 2015). Therefore, here, we demonstrate that the distributed reservoir-based forward models allows the robot to adapt the movements of a damaged leg and its walking gait, in order to deal with sudden leg damage situations. In this scenario, post learning of the forward models under the three different walking gaits, we initially let the robot walk with a tetrapod gait (**Figure 4B**, right). After 1000 steps (≈37 s) we constrained (deactivated) the FT-i joint (outermost) of the right middle leg such that the leg remains suspended in air and cannot achieve ground contact in this configuration. Thus, simulating leg damage scenario. AMOSII was then allowed to continue walking on the flat terrain under this damaged condition.

As observed in **Figure 9**, initially AMOSII walks under normal conditions (photo panel 1) with the right middle leg FT-i joint functioning normally. The FT-i joint was then constrained to 0° maximum and minimum angle of clearance (**Figure 9A**) thereby causing the right middle leg to be suspended in the air (photo panel 2). As a result the reservoir forward model prediction mismatches the current footcontact signal on the damaged leg, causing the accumulated error to gradually ramp up (**Figure 9D**). After a short transient period of AMOSII trying to walk in this configuration (dark green section in **Figure 9**), this results in adaptations in the FT-i and CT-i joints (yellow highlighted section in **Figures 9A,B**) thereby, allowing the robot to extend the damaged leg further down and support the locomotion (photo panels 3, 4, and 5). As a result, AMOSII was able to successfully keep walking straight with a slightly modified tetrapod gait despite the damaged right middle leg. Finally, after 2000 time steps (≈74 s), the FT-i joint was once again allowed to function normally, causing the accumulated error to become zero (the forward model prediction matches the actual footcontact signal). The robot then continues to walk as in the undamaged condition with a tetrapod gait. For further details, we encourage the readers to see the Supplementary Video 6 of the entire experiment. These results, thus clearly demonstrate that the distributed reservoir forward models not only allow complex locomotive behaviors, but also enable the

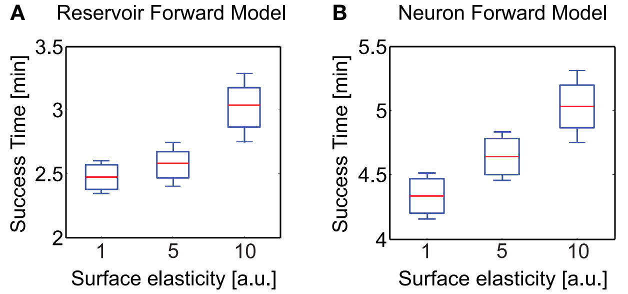


FIGURE 10 | Average time to successfully overcome uneven terrains of different elasticity (hard, moderate, highly elastic). (A) Average success time for reservoir-based forward model. **(B)** Average success time for adaptive neuron forward model from Manoonpong et al. (2013). Here the whiskers indicate one standard deviation above and below the mean value. Note the difference in scale of the y-axis in both plots. The experimental surface here consisted of the rough terrain as presented in **Figure 6** consisting of irregular undulations, however with varying degree of elasticity for the three cases.

robot to deal with unwanted changes in body properties in a robust manner.

In order to evaluate the performance of our adaptive reservoir forward model in comparison to the state of the art model recently presented in Manoonpong et al. (2013) (single recurrent neural with low-pass filter), we carried out simulation experiments with AMOSII walking on different types of surfaces. Specifically, after training on a flat surface (under normal conditions) we carried out 10 trials each with the robot walking on uneven terrains (laid with multiple obstacles of height 8 cm), having three different elastic properties[4]. The surfaces were divided into hard (1.0), moderately elastic (5.0) and highly elastic (10.0). A tetrapod walking gait was used in all three cases. Starting from a fixed position, we noted the total time taken by the robot to successfully cross the uneven terrain region and move into a flat surface region. As observed in **Figures 10A,B**, the reservoir forward model enables the robot to traverse the uneven region considerably faster as compared to the adaptive neuron forward model, in all three scenarios. Both the models can be seen to overcome the hard surface much better as compared to the elastic ones. This was expected due to the changes in surface stiffness resulting in additional forces on the robot legs. However, the reservoir model performance was considerably more robust with a mean difference in success time of 1.86 min for the hardest surface and approximately 2 min for the most elastic surface, cases. Given that the walking gait was fixed, here the success time can be thought as an indicator of the robot's energy efficiency. In the absence of additional body mechanisms to deal with changing surface stiffness, the reservoir based model outperforms the previous implementations of adaptive forward models by ≈25% on average. In the climbing and gap crossing scenarios, the performance of the two forward models are comparable (not shown here explicitly) unless there are significant changes in the ground reaction forces (e.g., climbing or crossing gaps on different types of terrain). As such the reservoir forward model offers a more generalized architecture for adaptive locomotion. Furthermore, as demonstrated previously, this model is also capable of robustly coping with missing motor information and a high degree of sensory noise; making use of the SARN internal memory and multiple timescales (Dasgupta, 2015). This was very difficult to achieve with the previous simple single recurrent neuron forward models. Moreover, the previous study also required that a separate forward model be learned for every different walking gait. Thus, creating a scalability issue for real robot implementations. Here, however, a single SARN can be trained online to predict the foot contact signals for multiple different walking gaits (here we show three gaits, but it can be easily extended to many more patterns—see Supplementary Figure 2, for tripod gait example).

5. Discussion

In this study, we presented adaptive forward models using the self-adaptive reservoir network for locomotion control. The model is implemented on each leg of a simulated bio-inspired hexapod robot. It is trained online during walking on a flat terrain in order to transform an efference copy (motor signal) into an expected foot contact signal (i.e., sensory prediction). Afterwards, the learned model of each leg is used to estimate walking states by comparing the expected foot contact signal with the actual incoming one. The difference between the expected and actual foot contact signals is used to adapt the robot's leg through elevation and searching control. Each leg is adapted independently. This enables the robot to successfully walk on uneven terrains. Moreover, using a backbone joint, the robot can also successfully cross a large gap and climb over a high obstacle as well as up a fleet of stairs. In this approach, basic walking patterns are generated by CPG-based control along with local leg control mechanisms that make use of the reservoir prediction to adapt the robot's behavior. The key neural mechanisms presented in this work, namely, CPG -based neural control, internal forward models and local leg control, are essential for robust, adaptive locomotion control. However, only individual instances of them has been successfully realized on artificial and bio-mimetic robotic systems (Bläsing, 2004; Pfeifer et al., 2007; Lewinger and Quinn, 2011; Ren et al., 2012; Schilling et al., 2012; Christensen et al., 2014; Cully et al., 2015); thereby achieving partial solutions. Furthermore, although a few studies have focused on a combination of these neural mechanisms, they have largely been tailored for adaptive locomotion in quadruped robots (Lewis and Bekey, 2002; Silva et al., 2012), without the ability to climb obstacles or cross large gaps, as observed in real animals and insects. Thus, this work demonstrates how the combination of these essential components, coupled with the power of the adaptive recurrent neural forward models can achieve very rich behavioral repertoire in bio-inspired hexapod robots. Thus, supporting the idea that such embodied neural control (Floreano et al., 2014) is indeed a potential powerful future alternative of more conventional control methods.

It is important to note that the usage of reservoir networks, as forward models here, provides the crucial benefit of an

[4] Here the elasticity coefficients do not strictly represent Young's modulus values. These were local parameter setting defined in the simulation, with increasing values causing greater elasticity.

inherent representation of time and fading memory (due to the internal feedback loops and input dependent adaptations). Such memory of the time-varying motor or sensory stimuli is required to overcome intrinsic time lags between expected sensory signals and motor outputs (Wolpert et al., 1998), as well as in behavioral scenarios with considerable dependence on the history of motor output (Lonini et al., 2009). This is very difficult in most of the previous implementations of forward internal models using either simple single recurrent neuron implementations (Manoonpong et al., 2013), feed-forward multi-layered neural networks (Schröder-Schetelig et al., 2010), or Bayesian network models (Dearden and Demiris, 2005; Sturm et al., 2008). Furthermore, in this case, online adaptation of only the reservoir-to-readout weights (readout) makes such networks beneficial for simple and online learning. The pre-training phase of the current setup was carried out only to gather sufficient statistics of the CTr-motor signals and foot-contact signals while walking under the different gaits, in order to learn the optimal reservoir neuron non-linearity and time constant parameters (Dasgupta et al., 2013). Subsequent to this, reservoir-to-readout weight learning occurs continuously without the need of any offline batch mode phase. Moreover, only a single learning trial under normal walking conditions was enough to learn the forward model for leg adaptations under different environmental situations. As a result making the reservoir based forward models very suitable for fast learning under real robot implementations.

The concept of forward models with efference copies in conjunction with neural control has been suggested since the mid-twentieth century (Holst and Mittelstaedt, 1950; Held, 1961) and increasingly employed for biological investigations (Webb, 2004). This is because it can explain mechanisms which biological systems use to predict the consequence of their action based on sensory information, resulting in adaptive and robust behaviors in a closed-loop scenario. This concept also forms a major motivation for robots inspired by biological systems. Within this context, the work presented here, verifies that a combination of CPG-based neural control, adaptive reservoir forward models with efference copies, and searching and elevation control can be used for robustly generating complex locomotion and adaptive behaviors in an artificial walking system. Additionally, although in this study we specifically focused on locomotive behaviors for walking robots, (such) SARN based motor prediction systems can be easily generalized to a number of other applications. Specifically for neuro-prosthetics (Ganguly and Carmena, 2009), sensor-driven orthotic control (Lee and Lee, 2005; Braun et al., 2014) or brain-machine interface devices (Golub et al., 2012), that require the learning of such predictive models using highly non-stationary, temporal signals, applying SARN models can provide high performance gains with embedded memory, as compared to the current static feed-forward neural network solutions.

In the future, we will transfer the reservoir-based adaptive forward models to the physical hexapod robot AMOS-II (Manoonpong et al., 2013) in order to test the adaptive behaviors in a real environment. Typically, the transfer of learning from simulation studies to physical hardware involves additional sensory and motor noise. As demonstrated in **Figures 5J,K**, the SARN based forward models are robust to significant levels of sensory noise as well as capable of dealing with corruption of motor signals. As such, although the currently presented results are in simulation, we envision that a transfer to a noisy real robot platform can be easily achieved. Furthermore, while the work presented here uses only a single CPG, the control mechanism and the distributed nature of the forward models allow for easy extension to multiple CPGs (Barikhan et al., 2014; Ren et al., 2015). For multiple CPGs, synchronization can emerge from continuous interactions of distributed CPGs, body dynamics, and the environment through local sensory feedback of each leg as shown in Barikhan et al. (2014); or can be also achieved by using a master-client mechanism with learning as demonstrated in our previous work (Ren et al., 2015).

Author Contributions

SD, FW, and PM designed the research. SD and PM implemented the model, analyzed data and carried out simulations. DG carried out the climbing experiments. SD and PM wrote the manuscript.

Acknowledgments

This research was supported by the Emmy Noether Program (DFG, MA4464/3-1), the Federal Ministry of Education and Research (BMBF) by a grant to the Bernstein Center for Computational Neuroscience II Göttingen (01GQ1005A, project D1) and the International Max Planck Research School for Physics of Biological and Complex Systems scholarship.

Supplementary Material

The Supplementary Material for this article can be found online at: http://journal.frontiersin.org/article/10.3389/fnbot.2015.00010

References

Barikhan, S. S., Wörgötter, F., and Manoonpong, P. (2014). "Multiple decoupled cpgs with local sensory feedback for adaptive locomotion behaviors of bio-inspired walking robots," in *From Animals to Animats 13* (Castellón: Springer), 65–75.

Beer, R. D., Chiel, H. J., Quinn, R. D., Espenschied, K. S., and Larsson, P. (1992). A distributed neural network architecture for hexapod robot locomotion. *Neural Comput.* 4, 356–365. doi: 10.1162/neco.1992.4.3.356

Berg, E. M., Hooper, S. L., Schmidt, J., and Büschges, A. (2015). A leg-local neural mechanism mediates the decision to search in stick insects. *Curr. Biol.* 25, 2012–2017. doi: 10.1016/j.cub.2015.06.017

Blaesing, B., and Cruse, H. (2004). Stick insect locomotion in a complex environment: climbing over large gaps. *J. Exp. Biol.* 207, 1273–1286. doi: 10.1242/jeb.00888

Bläsing, B. (2004). "Adaptive locomotion in a complex environment: simulation of stick insect gap crossing behaviour," in *From Animals to Animats 8* (Los Angeles, CA), 173–182.

Braun, J. M., Wörgötter, F., and Manoonpong, P. (2014). "Internal models support specific gaits in orthotic devices," in *Mobile Service Robotics, Number 17 in Proceedings of the International Conference on Climbing and Walking Robots* (Poznań), 539–546.

Christensen, D. J., Larsen, J. C., and Stoy, K. (2014). Fault-tolerant gait learning and morphology optimization of a polymorphic walking robot. *Evol. Syst.* 5, 21–32. doi: 10.1007/s12530-013-9088-3

Cruse, H. (1976). The control of body position in the stick insect (carausius morosus), when walking over uneven surfaces. *Biol. Cybern.* 24, 25–33. doi: 10.1007/BF00365591

Cruse, H., Kindermann, T., Schumm, M., Dean, J., and Schmitz, J. (1998). Walkneta biologically inspired network to control six-legged walking. *Neural Netw.* 11, 1435–1447. doi: 10.1016/S0893-6080(98)00067-7

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature* 521, 503–507. doi: 10.1038/nature14422

Dasgupta, S. (2015). *Temporal Information Processing and Memory Guided Behaviors with Recurrent Neural Networks*. Ph.D. thesis, Georg-August University, Göttingen.

Dasgupta, S., Wörgötter, F., and Manoonpong, P. (2013). Information dynamics based self-adaptive reservoir for delay temporal memory tasks. *Evol. Syst.* 4, 235–249. doi: 10.1007/s12530-013-9080-y

Dearden, A., and Demiris, Y. (2005). "Learning forward models for robots," in *International Joint Conference on Artificial Intelligence*, Vol. 5 (San Francisco, CA), 1440.

Der, R., and Martius, G. (2012). "The LPZRobots simulator," in *The Playful Machine* (Berlin; Heidelberg: Springer), 293–308. doi: 10.1007/978-3-642-20253-7_16

Floreano, D., Ijspeert, A. J., and Schaal, S. (2014). Robotics and neuroscience. *Curr. Biol.* 24, 910–920. doi: 10.1016/j.cub.2014.07.058

Ganguly, K., and Carmena, J. M. (2009). Emergence of a stable cortical map for neuroprosthetic control. *PLoS Biol.* 7:e1000153. doi: 10.1371/journal.pbio.1000153

Goldschmidt, D., Wörgötter, F., and Manoonpong, P. (2014). Biologically-inspired adaptive obstacle negotiation behavior of hexapod robots. *Front. Neurorobot.* 8:3. doi: 10.3389/fnbot.2014.00003

Golub, M. D., Yu, B., and Chase, S. M. (2012). "Internal models engaged by brain-computer interface control," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE* (San Diego, CA: IEEE), 1327–1330.

Held, R. (1961). Exposure-history as a factor in maintaining stability of perception and coordination. *J. Nerv. Ment. Dis.* 132, 26–32. doi: 10.1097/00005053-196101000-00005

Hesse, F., Martius, G., Manoonpong, P., Biehl, M., and Wörgötter, F. (2012). "Modular robot control environment testing neural control on simulated and real robots," in *Frontiers in Computational Neuroscience, Conference Abstract: Bernstein Conference* (Munich), 1416–1420.

Holst, E., and Mittelstaedt, H. (1950). Das reafferenzprinzip. *Naturwissenschaften* 37, 464–476. doi: 10.1007/BF00622503

Huston, S. J., and Jayaraman, V. (2011). Studying sensorimotor integration in insects. *Curr. Opin. Neurobiol.* 21, 527–534. doi: 10.1016/j.conb.2011.05.030

Ijspeert, A. J. (2014). Biorobotics: using robots to emulate and investigate agile locomotion. *Science* 346, 196–203. doi: 10.1126/science.1254486

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

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Curr. Opin. Neurobiol.* 9, 718–727. doi: 10.1016/S0959-4388(99)00028-8

Kesper, P., Grinke, E., Hesse, F., Wörgötter, F., and Manoonpong, P. (2013). Obstacle/gap detection and terrain classification of walking robots based on a 2d laser range finder. *Chapter* 53, 419–426. doi: 10.1142/9789814525534/0053

Lee, J.-W., and Lee, G.-K. (2005). Gait angle prediction for lower limb orthotics and prostheses using an emg signal and neural networks. *Int. J. Control Autom. Syst.* 3, 152–158.

Lewinger, W. A., and Quinn, R. D. (2011). Neurobiologically-based control system for an adaptively walking hexapod. *Ind. Robot. Int. J.* 38, 258–263. doi: 10.1108/01439911111122752

Lewis, M. A., and Bekey, G. A. (2002). Gait adaptation in a quadruped robot. *Auton. Robots* 12, 301–312. doi: 10.1023/A:1015221832567

Lonini, L., Dipietro, L., Zollo, L., Guglielmelli, E., and Krebs, H. I. (2009). An internal model for acquisition and retention of motor learning during arm reaching. *Neural Comput.* 21, 2009–2027. doi: 10.1162/neco.2009.03-08-721

Maass, W., Natschläeger, 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

Manoonpong, P., Parlitz, U., and Wörgötter, F. (2013). Neural control and adaptive neural forward models for insect-like, energy-efficient, and adaptable locomotion of walking machines. *Front. Neural Circuits* 7:12. doi: 10.3389/fncir.2013.00012

Mischiati, M., Lin, H.-T., Herold, P., Imler, E., Olberg, R., and Leonardo, A. (2015). Internal models direct dragonfly interception steering. *Nature* 517, 333–338. doi: 10.1038/nature14045

Nachstedt, T., Wörgötter, F., Manoonpong, P., Ariizumi, R., Ambe, Y., and Matsuno, F. (2013). "Adaptive neural oscillators with synaptic plasticity for locomotion control of a snake-like robot with screw-drive mechanism," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (Karlsruhe: IEEE), 3389–3395.

Pasemann, F., Hild, M., and Zahedi, K. (2003). "So(2)-networks as neural oscillators," in *Computational Methods in Neural Modeling*, eds J. Mira and J. R. Álvarez (Berlin: Springer), 144–151.

Pearson, K., and Franklin, R. (1984). Characteristics of leg movements and patterns of coordination in locusts walking on rough terrain. *Int. J. Robot. Res.* 3, 101–112. doi: 10.1177/027836498400300209

Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *Science* 318, 1088–1093. doi: 10.1126/science.1145803

Ren, G., Chen, W., Dasgupta, S., Kolodziejski, C., Wörgötter, F., and Manoonpong, P. (2015). Multiple chaotic central pattern generators with learning for legged locomotion and malfunction compensation. *Inf. Sci.* 294, 666–682. doi: 10.1016/j.ins.2014.05.001

Ren, G., Chen, W., Kolodziejski, C., Wörgötter, F., Dasgupta, S., and Manoonpong, P. (2012). "Multiple chaotic central pattern generators for locomotion generation and leg damage compensation in a hexapod robot," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on* (Vilamoura: IEEE), 2756–2761.

Schilling, M., Paskarbeit, J., Schmitz, J., Schneider, A., and Cruse, H. (2012). "Grounding an internal body model of a hexapod walker control of curve walking in a biologically inspired robot," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on* (IEEE), 2762–2768.

Schröder-Schetelig, J., Manoonpong, P., and Wörgötter, F. (2010). Using efference copy and a forward internal model for adaptive biped walking. *Auton. Robots* 29, 357–366. doi: 10.1007/s10514-010-9199-7

Silva, P., Matos, V., and Santos, C. P. (2012). "Adaptive quadruped locomotion: learning to detect and avoid an obstacle," in *From Animals to Animats 12* (Odense: Springer), 361–370.

Simon, H. (2002). *Adaptive Filter Theory,* Vol. 2. Englewood Cliffs, NJ: Prentice Hall.

Sompolinsky, H., Crisanti, A., and Sommers, H. (1988). Chaos in random neural networks. *Phys. Rev. Lett.* 61:259. doi: 10.1103/PhysRevLett.61.259

Steingrube, S., Timme, M., Wörgötter, F., and Manoonpong, P. (2010). Self-organized adaptation of a simple neural circuit enables complex robot behaviour. *Nat. Phys.* 6, 224–230. doi: 10.1038/nphys1508

Sturm, J., Plagemann, C., and Burgard, W. (2008). "Adaptive body scheme models for robust robotic manipulation," in *Robotics: Science and Systems* (Zurich).

Sussillo, D., and Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron* 63, 544–557. doi: 10.1016/j.neuron.2009.07.018

Triesch, J. (2005). "A gradient rule for the plasticity of a neurons intrinsic excitability," in *Artificial Neural Networks: Biological Inspirations–ICANN 2005* (Warsaw: Springer), 65–70.

van Vreeswijk, C., and Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* 274, 1724–1726. doi: 10.1126/science.274.5293.1724

Watson, J. T., Ritzmann, R. E., Zill, S. N., and Pollack, A. J. (2002). Control of obstacle climbing in the cockroach, blaberus discoidalis. i. kinematics. *J. Compa. Physiol. A* 188, 39–53. doi: 10.1007/s00359-002-0277-y

Webb, B. (2004). Neural mechanisms for prediction: do insects have forward models? *Trends Neurosci.* 27, 278–282. doi: 10.1016/j.tins.2004.03.004

Wolpert, D. M., Miall, R. C., and Kawato, M. (1998). Internal models in the cerebellum. *Trends Cogn. Sci.* 2, 338–347. doi: 10.1016/S1364-6613(98)01221-2

Zenker, S., Aksoy, E. E., Goldschmidt, D., Wörgötter, F., and Manoonpong, P. (2013). "Visual terrain classification for selecting energy efficient gaits of a hexapod robot," in *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on* (IEEE), 577–584.

Zill, S., Schmitz, J., and Büschges, A. (2004). Load sensing and control of posture and locomotion. *Arthropod Struct. Dev.* 33, 273–286. doi: 10.1016/j.asd.2004.05.005

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.