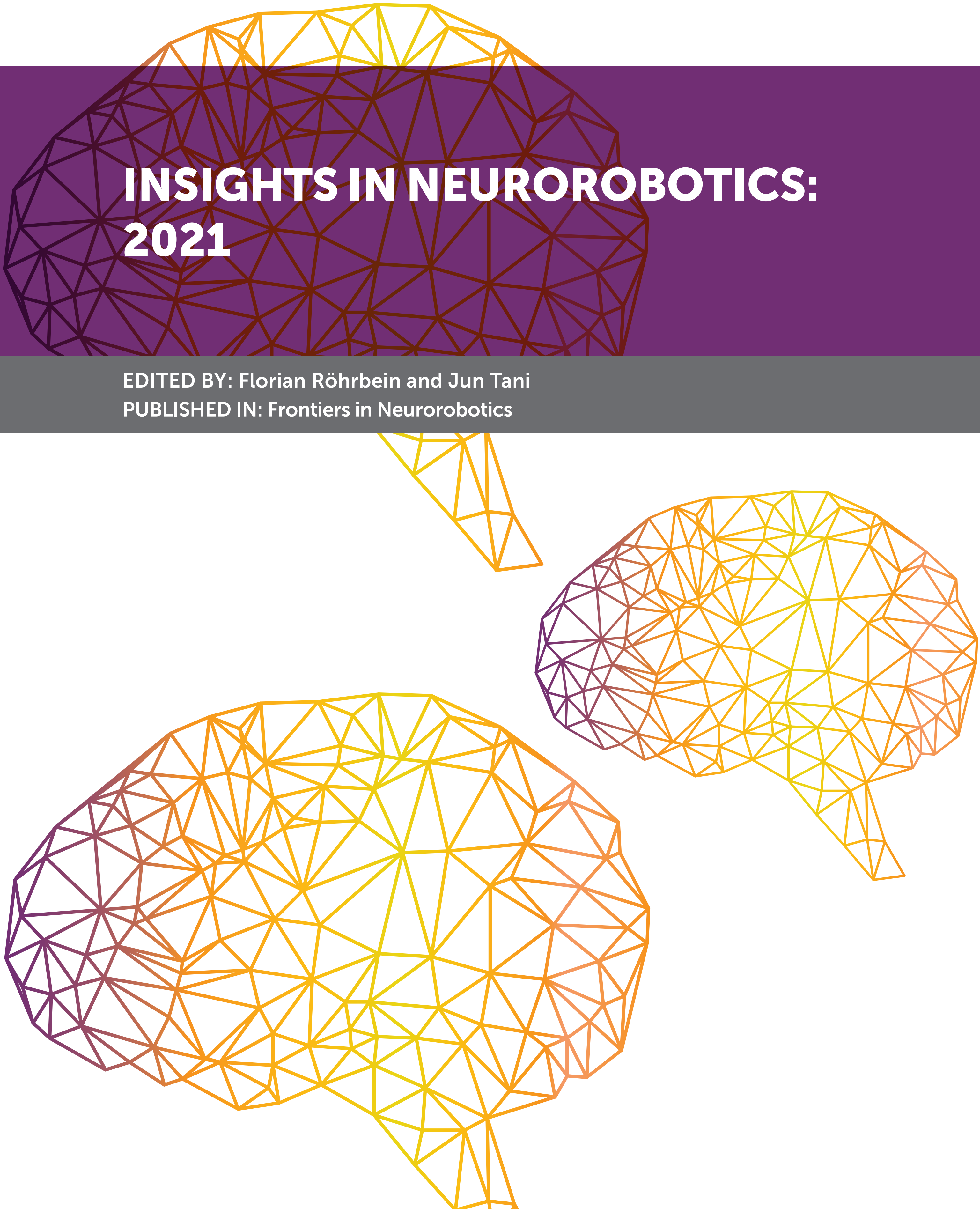
INSIGHTS IN NEUROROBOTICS: 2021

EDITED BY: Florian Röhrbein and Jun Tani
PUBLISHED IN: Frontiers in Neurorobotics



frontiers Research Topics

ISSN 1664-8714
ISBN 978-2-83250-590-8
DOI 10.3389/978-2-83250-590-8

INSIGHTS IN NEUROROBOTICS: 2021

Topic Editors:
Florian Röhrbein, Technische Universität Chemnitz, Germany
Jun Tani, Okinawa Institute of Science and Technology Graduate University, Japan

Citation: Röhrbein, F., Tani, J., eds. (2022). Insights in Neurorobotics: 2021. Lausanne: Frontiers Media SA. doi: 10.3389/978-2-83250-590-8

Table of Contents

TECHNOLOGY AND CODE
published: 25 April 2022
doi: 10.3389/fnbot.2022.856727

The Neurorobotics Platform Robot Designer: Modeling Morphologies for Embodied Learning Experiments

Benedikt Feldotto*, Fabrice O. Morin and Alois Knoll

Robotics, Artificial Intelligence and Real-Time Systems, Faculty of Informatics, Technical University of Munich, Munich, Germany

The more we investigate the principles of motion learning in biological systems, the more we reveal the central role that body morphology plays in motion execution. Not only does anatomy define the kinematics and therefore the complexity of possible movements, but it now becomes clear that part of the computation required for motion control is offloaded to body dynamics (a phenomenon referred to as "Morphological Computation.") Consequentially, a proper design of body morphology is essential to carry out meaningful simulations on motor control of robotic and musculoskeletal systems. The design should not be fixed for simulation experiments beforehand, but is a central research aspect in every motion learning experiment that requires continuous adaptation during the experimental phase. We herein introduce a plugin for the 3D modeling suite Blender that enables researchers to design morphologies for simulation experiments in, particularly but not restricted to, the Neurorobotics Platform. We include design capabilities for both musculoskeletal bodies, as well as robotic systems in the Robot Designer. Thereby, we hope to not only foster understanding of biological motions and enabling better robot designs, but enabling true Neurorobotic experiments that may consist of biomimetic models such as tendon-driven robot as a mix of both or a transition between both biology and technology. This plugin helps researchers design and parameterize models with a Graphical User Interface and thus simplifies and speeds up the overall design process.

Keywords: simulation, neurorobotics, design, biomimetic robots, muscles, biomechanics, embodied AI

OPEN ACCESS

Edited by:
Matej Hoffmann,
Czech Technical University in Prague, Czechia

Reviewed by:
Emilio Cartoni,
National Research Council (ISTC-CNR), Italy
Silvio Traversaro,
Italian Institute of Technology (IIT), Italy

****Correspondence:***
Benedikt Feldotto
feldotto@in.tum.de

Received: *17 January 2022*
Accepted: *23 March 2022*
Published: *25 April 2022*

Citation:
Feldotto B, Morin FO and Knoll A (2022) The Neurorobotics Platform Robot Designer: Modeling Morphologies for Embodied Learning Experiments. Front. Neurorobot. 16:856727. doi: 10.3389/fnbot.2022.856727

1. INTRODUCTION

The term Morphological Computation (Müller and Hoffmann, 2017) describes the principle of computations required for motion execution that are not implemented in a dedicated (electronic) controller but executed by the kinematics and morphology of the body itself. A well studied example for morphology facilitating control can be found in the passive dynamic walker (McGeer, 1990; Müller and Hoffmann, 2017), that locomotes smoothly down a slope without any computational unit and no energy input except gravitation. This principle posits that body morphology not only contributes to the execution of motions, but also that body design is a crucial determinant of behavioral capabilities of a given agent. Furthermore, several studies suggest that the morphological design may even have a decisive effect on cognition and hereby may influence thinking and problem solving (Pfeifer, 2000; Pfeifer and Bongard, 2006). At the very least, kinematics, dynamics, and geometry / anatomy enable or constrain actions that an agent is able to execute, as they mediate any intelligent interaction of an agent with its surrounding world. In a robotic view, the kinematic structure defines the workspace, actuators and sensors enable interaction with the environment in terms of action and perception.

The aforementioned constraints, computations, and influence of the morphology that impact the overall agent capabilities emphasize the importance of proper and thoughtful body design for (energy-) efficient and functionally capable agents. Therefore, we herein introduce the Robot Designer, a plugin for the 3D modeling suite Blender (Community, 2018) to facilitate the design of musculoskeletal, as well as robotic body models for simulation-based experiments. We propose a graphical user interface (GUI) with a range of tools for kinematics, dynamics, geometries, sensors, and muscles to promote easy and fast design and parameterization of agent bodies. Additionally, environmental setups can be generated. Models can be exported/imported in community standard formats such as SDFormat and .osim and are directly compatible with the Neurorobotics Platform (NRP) as a framework for embodied motion learning experiments. With the Robot Designer we introduce the first morphology design tool that integrates capabilities for both biological and robotic morphologies.

The Robot Designer is an ongoing Open Source project published under GPLv2 license in the BlenderRobotDesigner repository [1] of the Neurorobotics group in the Human Brain Project. Research in Artificial Intelligence and neural learning in the brain can largely benefit from this plugin as it helps building virtual environments for training data collection in simulation as well as opening the possibility to connect *in silico* brain models to musculoskeletal bodies, that cannot be built in the physical world, respectively.

In this article, we will introduce the State of the Art in robot design tools, and the Neurorobotics Platform as the central simulation as a prominent example on Neurorobotics simulation suite. We will then introduce the architecture and workflow building robot models with the Robot Designer before we go in depth into the various design steps and capabilities. Finally, we will present several examples of models created with the help of our plugin including robotic, musculoskeletal and biomimetic systems.

2. STATE OF THE ART

Simulation of the physical world has become ever more sophisticated in the past years in terms of computation of complex kinematic chains, computation performance for large 3D environments and photorealistic rendering. Improvements in simulation software have been matched by a strong increase in computational power available locally or on demand in highly parallel cloud computing instances. Robotic Machine Learning applications requiring large amounts of training data have grown rapidly and therefore robot simulation has attracted great interest while simultaneously stimulating the expression of many requirements.

Several simulation suites exist that are build for dedicated purposes; a comprehensive review of physics simulators can be found in Collins et al. (2021). The design and parameterization of virtual agents and environments, however, remains a tedious and time-consuming process. This is one reason why models and environments are usually provided within benchmark experiments and often are not modified by the actual users after release. Popular examples can be found in the field of Artificial Intelligence with the OpenAI gym environments (Brockman et al., 2016) and the NIPS learning to run challenge (Brockman et al., 2016).

Working with predefined models and environments eliminates the ability for users to exploit the full potential of intelligent body design that itself supports fast and efficient learning. In order to offer guided model design and easier parameterization while at the same time enabling a co-design process of morphology model along with the cognitive module, user-friendly design tools are necessary.

In this chapter, we introduce existing tools and conventions for simulation model design that we utilize for the Robot Designer or are related to our work. We also present the Neurorobotics Platform as an embodied learning suite wherein our exported models can be used.

2.1. Related Tools

The most basic and yet popular approach for model parameterization still is editing respective values in the model XML description. In order to generate a new model from scratch, one usually uses mesh files (commonly .dae or .stl file format) that originate from CAD software (e.g., CATIA, Autodesk, SolidWorks), or mesh generation with 3D modeling tools such as Blender and Maya, or other editing tools that specialize on specific model types (e.g., MakeHuman for human avatars). For some CAD design tools the community has developed plugins that support SDF or URDF (can be converted to SDF using Gazebo) export functionalities [e.g., SDFusion for AutoDesk Fusion 360 (Roboy development team, 2021), FreeCAD RobotCreator Workbench (Fosselius, 2017) and sw_urdf_exporter for SolidWorks (StephenBrawner, 2021)]. Most of them are inofficial though, and parameters that go beyond geometry and kinematics still need to be added manually later on. Many simulators include GUI elements to inspect or edit model parameters (e.g., Gazebo, Opensim). Gazebo also includes a dedicated model editor to build up models using existing mesh files or adapting existing models. Usually, a mix of manual XML editing, use of specialized tools for mesh editing, and use of available simulator user interfaces is utilized, and users build up their own custom pipelines with a mix of preferred tools.

The tool most similar to our plugin is the Phobos project (von Szadkowski and Reichel, 2020), that combines robot model parameterization, as well as sophisticated mesh editing in a single framework and is also implemented as a Blender plugin. The graphical user interface is optimized for robotic design, users can adapt all relevant parameters *via* GUI elements, and models

[1] https://github.com/HBPNeurorobotics/BlenderRobotDesigner

can be exported/imported in URDF, SDF, and SMURF format. Phobos implements similar functionalities as the Robot Designer, the main differences being that we split the GUI in multiple tabs that represent and hereby structure the design process, as well as our focus on Neurorobotic models with support for musculoskeletal simulations.

The first iterations of the NRP RobotDesigner were initially developed under the name OpenGRASP RobotEditor (Leon et al., 2010) from 2008-2012 at the Humanoids and Intelligence Systems Lab (HIS) at the Institute for Anthropomatics and Robotics (IAR) of the Karlsruhe Institute of Technology in the context of the European GRASP project (Leon et al., 2010) as part of the OpenGRASP software.

The RobotEditor software focused on robotic design capabilities, as the support for valid models in Collada v1.5 data 3D asset exchange format had large support from the industry. It was later rewritten to comply with the programming interface of newer Blender versions (>2.69); support for sensors matching motion capture data for the Simox robot simulator (Vahrenkamp et al., 2012) was also added. The Robot Designer was initially forked from the RobotEditor project. Many features were added or optimized; in particular its main focus was shifted from pure robotic to Neurorobotic model design and direct compliance with the NRP.

2.2. Blender

Blender (Community, 2018) is an open source and feature rich suite for 3D modeling licensed under GPL. It includes tools for modeling, animation, rendering, compositing and motion tracking, video editing and 2D animation. Additionally, models can be rigged and it integrates a physics engine for simulation. A major strength of Blender is its open Python programming interface with integrated terminal and scripting that allows users to automate model design and easily write custom scripts and plugins. Blender provides all necessary tools for 3D mesh design suitable for Neurorobotic models. The Robot Designer extends the build in datatypes for robot and musculoskeletal modeling and its import and export. The plugin is integrated into Blenders plugin framework and is regularly updated to newer blender software versions, currently Blender version 2.82. **Figure 1** shows the Graphical User Interface of Blender 2.82, on the right you can see the expanded Robot Designer Plugin that is subdivided into multiple tab sessions.

2.3. SDFormat

The Simulation Description Format (SDF) (Open Source Robotics Foundation, 2020) is a standardized XML description for models and environments in 3D simulations. It is specifically designed for robotic simulations that include environments, objects, and robot actors. The description format specifies kinematics, dynamics, and geometries of models that can be extended with sensors, actuators, and animations. General properties of the physics engine, environmental conditions, such as gravity and lighting and many others parameters can be specified; one or multiple passive objects and active robotic agents can be integrated. Mesh configurations can be linked as Collada (.dae) or STL files. SDFormat is particularly used for, but not limited to, the robotic simulator Gazebo (Koenig and Howard, 2004) that is widely used in the robotic community.

2.4. The Neurorobotics Platform

The Neurorobotics Platform (Knoll et al., 2016; Falotico et al., 2017; Albanese et al., 2020) is a simulation framework for embodied neural simulations that is developed in the framework of the Human Brain Project. At its core, spiking neural networks can be interconnected *via* so-called Transfer Functions with robotic or musculoskeletal simulation models interacting in virtual environments. With this approach, learning in neural networks can be studied in the context of a closed Perception-Cognition-Action loop (Vernon et al., 2015). The open source platform itself is built upon existing community tools, such as NEST (Gewaltig and Diesmann, 2007) for spiking neural networks, Gazebo (Koenig and Howard, 2004) and OpenSim (Delp et al., 2007) for body physics simulation, and ROS (Stanford Artificial Intelligence Laboratory et al., 2021) as communication interface. The platform can be installed locally but is also offered as a service on supercomputing cluster resources *via* browser login. Overall the platform aims to support understanding of embodiment for neural learning. Biologically plausible or artificial neural networks can be easily interfaced with musculoskeletal systems and robotic models, thereby supporting the study of transfer learning from biological to artificial agents and vice versa. The platform comes with a set of tools to implement, run and analyze embodied learning experiments: Robot and Environment Designer, browser-based Graphical User Interface for interactive experiment execution, plotting and spike train widgets for analysis, as well as a Virtual Coach for scripted (batch) experiment execution.

The Robot Designer has been introduced as a tool alongside this suite of tools within the NRP in (Falotico et al., 2017) in order to design and optimize simulation models. Since its introduction many improvements and adaptations have been made, in particular the support for biomimetic and musculoskeltal models.

In this article, we describe the Robot Designer in detail and introduce the evolved architecture, workflow and tools. Our plugin has been tested with NRP version 3.1 and will be continuously adapted for compatibility with future versions.

3. PLUGIN ARCHITECTURE

The Robot Designer is implemented as a plugin in the 3D modeling suite Blender. Thereby, the capabilities of Blender can be utilized in order to design, e.g., geometries and add textures. At the time of writing of the present report, the supported version is Blender 2.8; continuous improvements will adapt the plugin for future releases. The installation instructions and a respective installation script for the Robot Designer in Blender can be found in the Readme description of the referenced GitHub repository. Descriptions in this article refer to the Robot

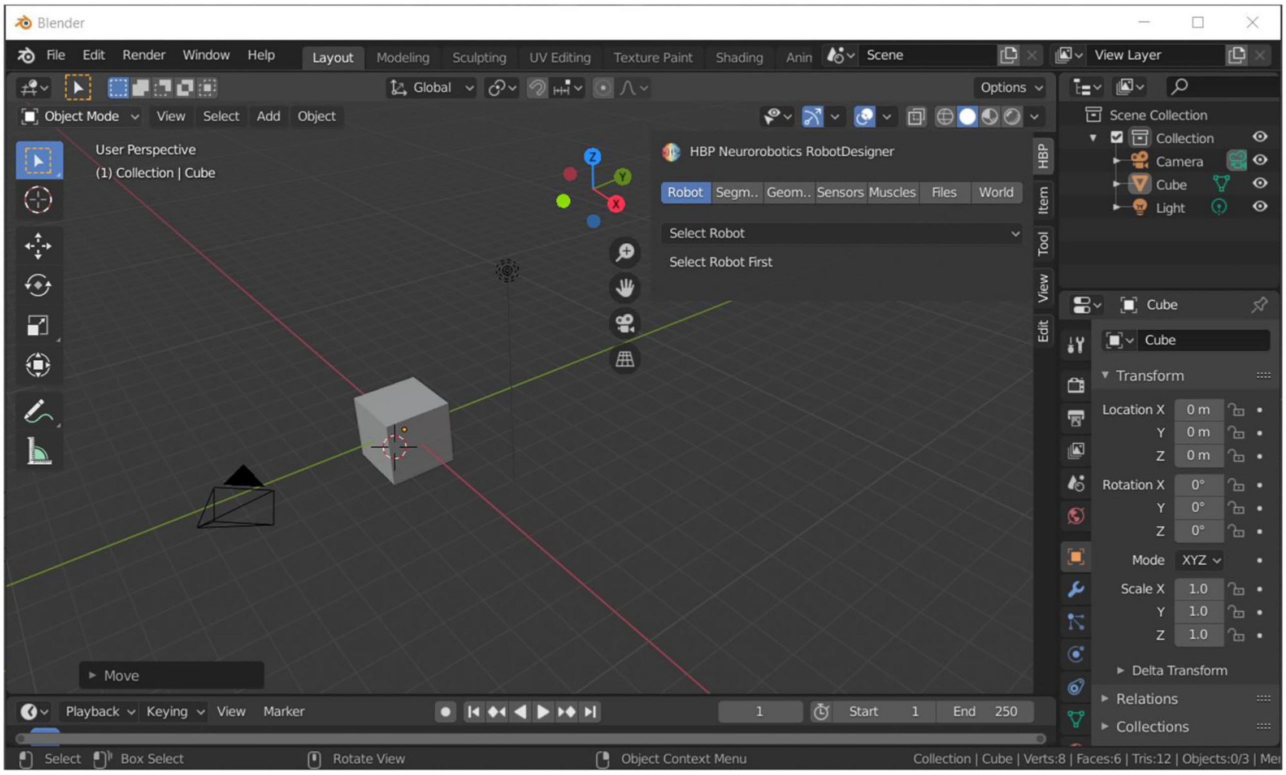

Blender
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Anin
Scene
View Layer
Global
Options
Object Mode View Select Add Object
User Perspective
(1) Collection | Cube
HBP Neurorobotics RobotDesigner
Robot Segm.. Geom.. Sensors Muscles Files World
Select Robot
Select Robot First
HBP Item Tool View Edit
Scene Collection
Collection
Camera
Cube
Light
Cube
Transform
Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ
Scale X 1.0
Y 1.0
Z 1.0
Delta Transform
Relations
Collections
Move
Playback Keying View Marker
1 Start 1 End 250
Select Box Select Rotate View Object Context Menu
Collection | Cube | Verts:8 | Faces:6 | Tris:12 | Objects:0/3 | Mer

FIGURE 1 | Graphical User Interface (GUI) of the 3D modeling suite Blender. The Robot Designer is implemented as a plugin to extend the design tools for robot and musculoskeletal modeling, specifically it can be expanded from the menu as shown in the top-right area of the figure.

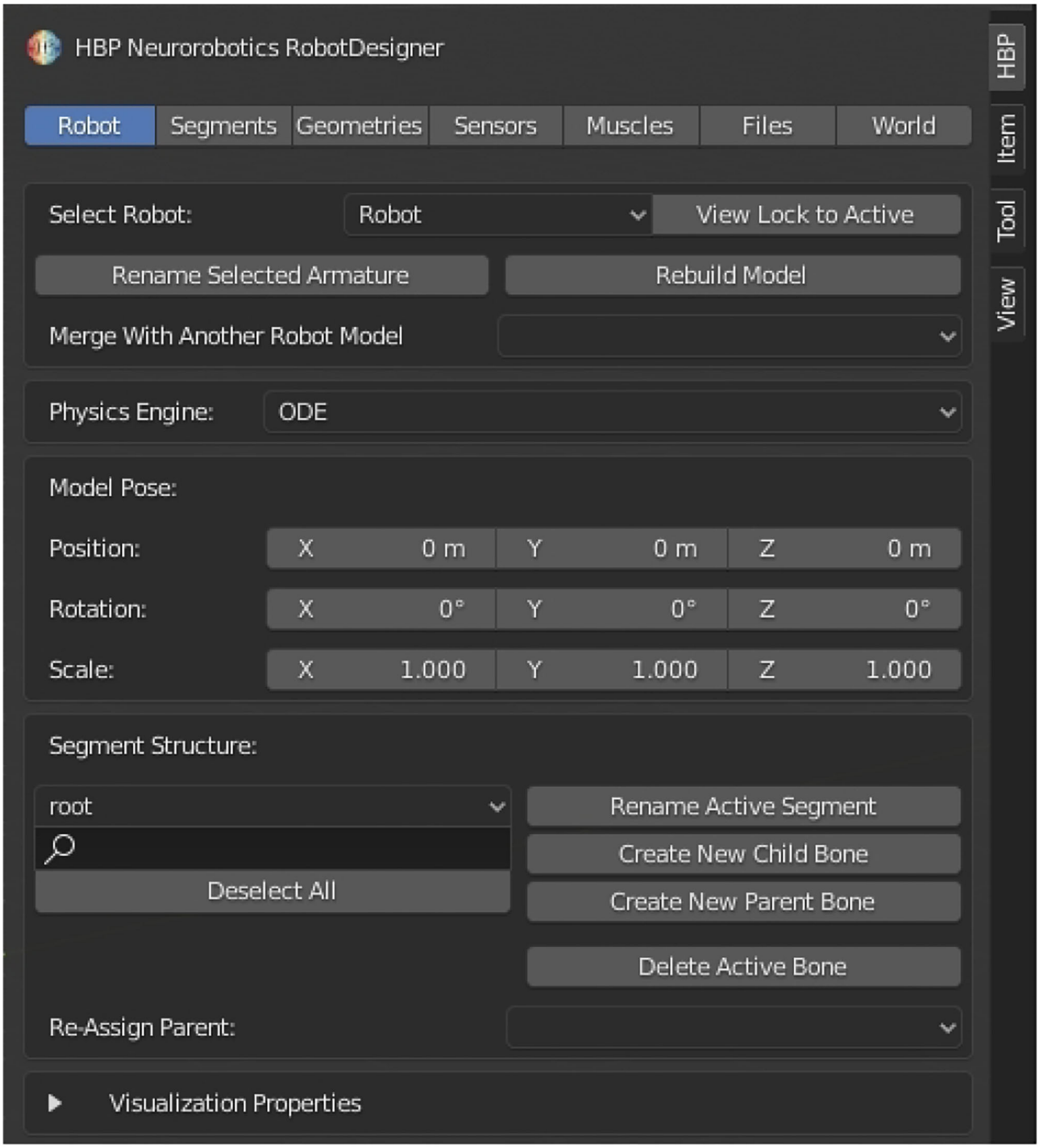
Designer plugin version 3.2, additional screenshots from Robot Designer 3.1 (Blender 2.7) demonstrate the consistent workflow we establish with ongoing plugin development. With the Robot Designer models can be extended with robot and musculoskeletal specific characteristics. Properties include defining kinematics, dynamics and geometries, adding sensors and muscles and importing/exporting models in simulator-specific formats. With this integrative setup, users have the ability to build models from scratch by designing meshes and then building up the kinematic structure within a single framework. **Figure 2** shows the graphical user interface of the Robot Designer Plugin. The user is guided through the design process by section tabs on the top from left to right. The Robot Designer is built upon Blender built-in functionalities and parameters where possible. In particular, it uses the built in rigging functionalities and extends and adds parameters as well as tools where necessary.

The plugin itself is structured into the following sub-folders, as it has been introduced in the RobotEditor project:

- core.
- export.
- interface.
- operators.
- properties.
- resources.

The listed components contain: Implementations for plugin registration ("core,") import and export functionalities ("export,") the graphical user interface ("interface,") operator functions that implement the various design tools of the Robot Designer ("operators,") global and specific parameters ("properties,") as well as additional resources (such as a created logfile) ("resources.") Generally, the plugin is separated in frontend (interface) and backend (operators) functionalities. Inside the "properties" folder, global properties, as well as separate files holding properties specific to objects and segments exist. Properties are registered to the scene (*global) or to dedicated Blender objects, respectively. Hereby, existing Blender objects are extended in our plugin with parameters specific to robot or musculoskeletal modeling.

The Robot Designer "core" implements a general plugin manager that is responsible for registration of the plugin in Blender, as well as operators and properties to the Blender namespace. Additionally, it holds the general plugin configuration. The "core" is mostly original from the RobotEditor and implements base classes for the graphical user interface, in particular a collapsible box and info box template. Similarly, a base class for operators is implemented that adds pre- and postconditions implemented as decorator functions. Through this mechanism, proper function of operators is ensured, e.g. the given function is only executed if a specific object



Select Robot: Robot
Merge With Another Robot Model
Physics Engine: ODE
Model Pose:
Position: X 0 m Y 0 m Z 0 m
Rotation: X 0° Y 0° Z 0°
Scale: X 1.000 Y 1.000 Z 1.000
Segment Structure: root
Re-Assign Parent:

FIGURE 2 | Robot Designer Graphical User Interface: The user interface is subdivided into separate sections, which guide the user through the dedicated steps in the design process and can be selected *via* tabs on the top line. Sections include: Robot, Segments, Geometries, Sensors, Muscles, Files, World.

is selected that is to be modified e.g. an armature when editing the robot or a mesh editing a geometry. A property handler for single properties and property groups wraps factory functions, adding not only getter and setter functions but also search functionalities. Lastly, the core implements logging to a logfile, adding functions that retrieves and formats the callstack.

For the plugin we make use of as most Blender properties, types and data objects as possible. In many cases, Blender objects are utilized to maintain common 3D modification and visualization using the Blender tools. Re-using meshes and curves as Robot Designer objects we add a property to the object that contains a tag indicating our intended purpose and thus can easily search for and sort objects in the Robot Designer.

4. MODEL DESIGN WORKFLOW

With the Robot Designer, existing robot models can either be imported and adapted, or new models fully created from scratch. The general workflow is depicted in **Figure 3** and is implemented in various tabs of the graphical user interface: users are guided through the design process and can walk through it selecting the offered tabs from left to right. Models are created with geometry meshes that can be imported from e.g. CAD software or manually designed in Blender utilizing the numerous modeling tools available. By utilizing elements of the plugins Graphical User Interface, links, joints, dynamics, controllers (Section 5), geometries, collisions (Section 6), and sensors (Section 7) can all be defined. For musculoskeletal models, a dedicated muscle tab helps define and parameterize muscle objects (Section 8).

After adding model metadata in the "files" section of the GUI, a model folder can be exported of the model description (model.config), model definition according to SDFormat (model.sdf), muscle definition in OpenSim specification (muscles.osim), as well as the referenced collision and visual meshes. The robot model can be directly imported into the NRP and simulated; without muscles, the exported model can also be used in every simulator that can handle the SDF format.

On the bottom part of **Figure 4** a potential workflow is shown that goes from CAD design *via* Robot Designer parameterization to the final model simulated in the Neurorobotics Platfom. A video tutorial showcasing the design of a basic model as well as the graphical definition of muscles on a skeleton can be found in Feldotto (2017). Additionally, world files can be generated with environmental parameters for gravity and lighting, as well as one or multiple (robot) model instances (see Section 9).

5. RIGID BODIES

New model instances can be created, and existing instances selected, in the "Robot" tab. Here, the general 3D location, supported physics engine and the composition of links is defined. Afterwards, in the "Segments" section, kinematics, dynamics and controllers are specified as follows:

- **Kinematics**
  Every model consists of one or multiple segments that are interconnected *via* fixed or moving joints. This kinematic chain can be configured either in Euler angles $[x, y, z, \alpha, \beta, \gamma]$ (Euler, 1968) or according to the Denavit-Hartenberg convention $[\theta, d, \alpha, a]$ (Hartenberg and Denavit, 1964). Users can assign the desired joint type from a list of types such as fixed, prismatic or revolute, while static objects may be connected to the world with a fixed joint.
- **Dynamics**
  Physics properties can be specified as mass, center of mass and inertia matrix individually for every segment. If the material making up the links can be taken to be homogeneous and its density can be provided, the center of mass and inertia can be calculated automatically based on the mesh geometry. Additional dynamic parameters include static and dynamic friction coefficients.
- **Controllers**
  The Robot Designer supports PID controllers with proportional, integral and derivative parameters for every joint. These control parameters are exported as a general controller plugin shipped with the NRP in order to control every joint individually *via* ROS interface.

For every model selected, the kinematic chain is visualized as a linked graph in the 3D scene and inertia boxes can be visualized as translucent purple boxes, like in Gazebo. For the implementation of robotic links, we rely on the rigging functionalities of Blender: the kinematic chain is implemented as Blender armature (bpy.types.Armatures), that is extended with robot specific properties such as the Euler and Denavit-Hartenberg parameters. Inertia boxes are empty Blender objects for visualization and easy manipulation in the scene, again extended with the inertia matrix and weight properties.

6. GEOMETRIES

Next to the model kinematics, geometry plays an essential role, not only for visualization, but first and foremost in the computations of the interactions of a model with its surrounding environment. With the Robot Designer plugin all model geometries are implemented as Blender meshes that are tagged with an extended property in order to indicate their role for the given model. With the "Geometries" section tab the user can adjust the general properties for location, orientation, scaling and attach/detach geometries to the respective model links. For most physics engines, such as Gazebo, two separate geometries can be assigned per link. Collision meshes used for collision detection by the physics engine are typically a simplified version of the visual geometry; any increase in the complexity of these collision meshes entails an increase in computation time. The Robot Designers provides GUI buttons to enable easy addition of basic collision shapes (box, cylinder, sphere), as well as autogeneration of the convex hull of any visual geometry. Additionally, the vertex count per mesh can be reduced easily *via* a GUI button, for a single or all model meshes, in order to adjust the tradeoff of simplification and granularity for optimal physics engine performance and model behavior, respectively. In order to speed up the design process, many tools can be applied either to a single geometry or to all geometries of the selected model instance.

7. SENSORS

Complementary to the control and actuation of body limbs, agents are endowed with perceptual capabilities through a variety of sensors. Here, the Gazebo and ROS communities offer a wide variety of sensor plugins that can be added to the model SDF description. Within the Robot Designer "Sensors" section, sensor

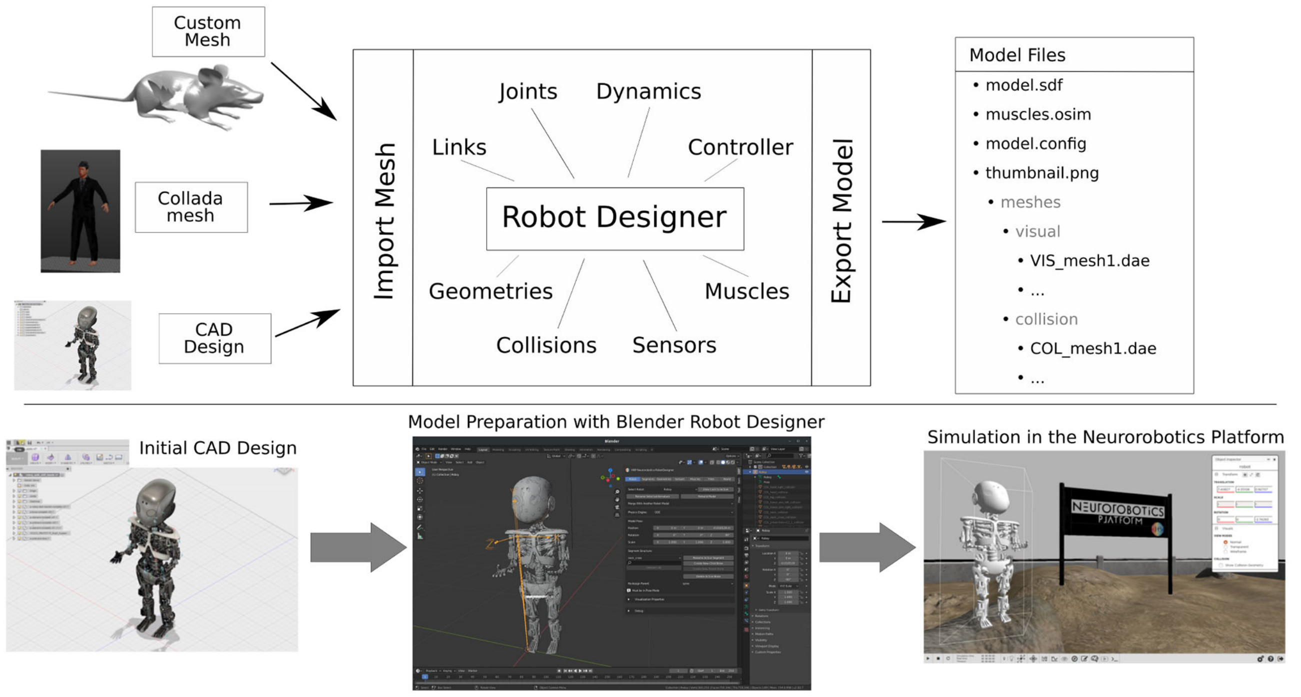


FIGURE 3 | Robot Designer Workflow: The user can create a new model either by designing custom meshes from scratch or by importing an existing geometry, e.g., from CAD software. With the Robot Designer, all specifications relevant for robots and musculoskeletal models can be added and modified, ranging from kinematics, dynamics and geometries to sensors and muscles. Finally, the model description files are exported. The bottom part visualizes an exemplary workflow from CAD design *via* model preparation with the Robot Designer in Blender and finally its simulation in the Neurorobotics Platform.

instances can be added to the model graphically by clicking the desired 3D position in the scene. Afterwards the sensor is added to a specific model link, and sensor-specific parameters can be adjusted in the GUI. Camera sensors build up on Blender cameras, and are extended with specific properties that can be specified in the SDFormat. Since all sensor and controller plugins are specified as (ROS-) Gazebo plugins, new types and customized plugins can later be easily added in the introduced framework. Support for import and export of sensors and plugins will be implemented soon.

8. MUSCLES

The Neurorobotics Platform integrates the framework for muscle dynamics and pathpoint simulation from OpenSim into the "Simbody" physics engine in Gazebo in order to enable simulation of musculoskeletal systems and "classical" rigid-body robots within the same environment. Additionally, this integration allows the simulation of biomimetic robots that exhibit some degree of mechanical compliance, e.g., joints connecting rigid links but controlled with muscle-like actuators such as the Myorobotics framework (Marques et al., 2013). The Robot Designer provides graphical tools to easily define pathways and parameters for muscle or muscle-like actuators, thus enabling future research on biomechanics, soft robotics, etc.

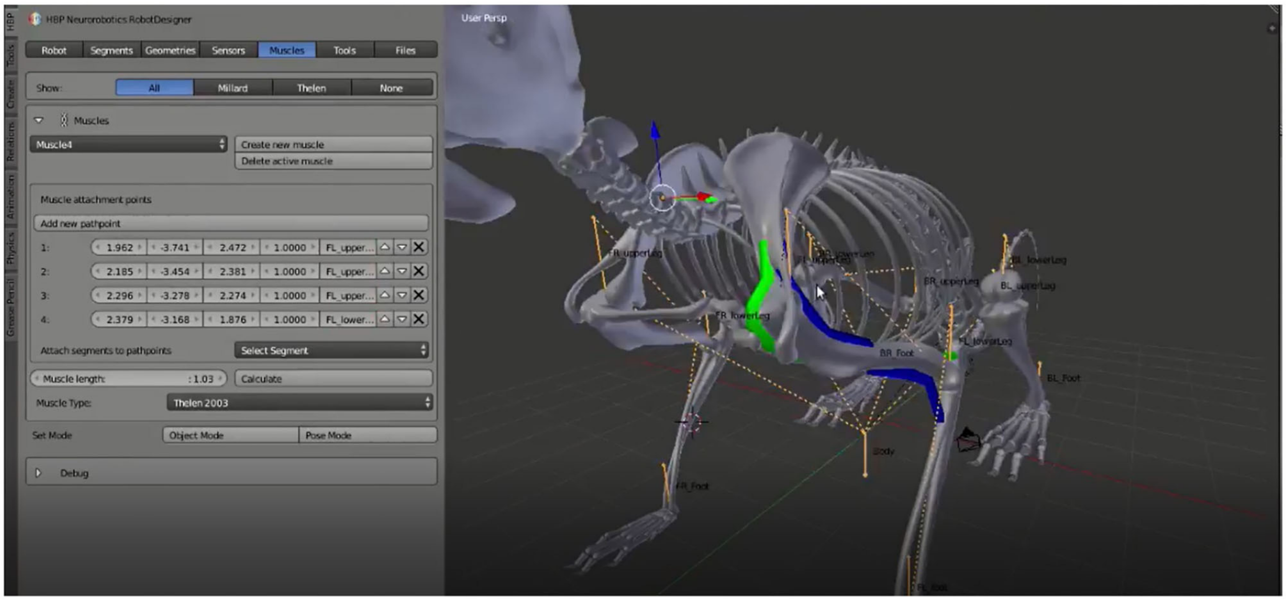
8.1. Path and Attachment Points

An arbitrary number of muscles can be defined as a set of pathpoints (attachment points or *via*-points defining the muscle routing). Pathpoints are added by selecting a 3D position with the mouse cursor and approving with a GUI button. The exact location can afterwards be refined using the listed coordinates, and the connected link selected *via* drop-down menu. For this graphical manipulation, we implemented every muscle as Blender curve data object that is visualized in the 3D scene. Every muscle pathpoint is defined as a point on the underlying spline and can therefore be edited using the common 3D tools. As such, a muscle is a curve object, tagged with a custom property as a Robot Designer muscle and extended with specific properties such as muscle type and respective parameters to be modified in the GUI. An operator is implemented in order to calculate the length of the muscle automatically *via* a dedicated button.

Figure 4 shows muscles defined on a biological rodent skeleton, as well as the corresponding graphical user interface with generated muscle pathpoints and chosen parameters. Color coding indicates the various muscle types.

8.2. Wrapping Surfaces

OpenSim supports wrapping surfaces for muscles that do not follow a straight line but rather adapt and wrap to the skeletal surface. In the "muscles" section of the GUI, basic shapes (sphere, cylinder) can be added to the scene, attached to a segment and parameterized accordingly, whereas the process of mesh



Muscle4

1:	1.962	-3.741	2.472	1.0000	FL_upper...
2:	2.185	-3.454	2.381	1.0000	FL_upper...
3:	2.296	-3.278	2.274	1.0000	FL_upper...
4:	2.379	-3.168	1.876	1.0000	FL_lower...

Muscle length: : 1.03

Muscle Type: Thelen 2003

FIGURE 4 | Musculoskeletal Model Design: Muscles are defined for a biological rodent model. Using the GUI and scene interaction, muscles can be specified with an arbitrary number of pathpoints by clicking the desired locations directly on the 3D model. All generated points are listed in the GUI and the exact location can be adjusted. Afterwards, muscle parameters such as muscle type, length and maximal force can be specified.

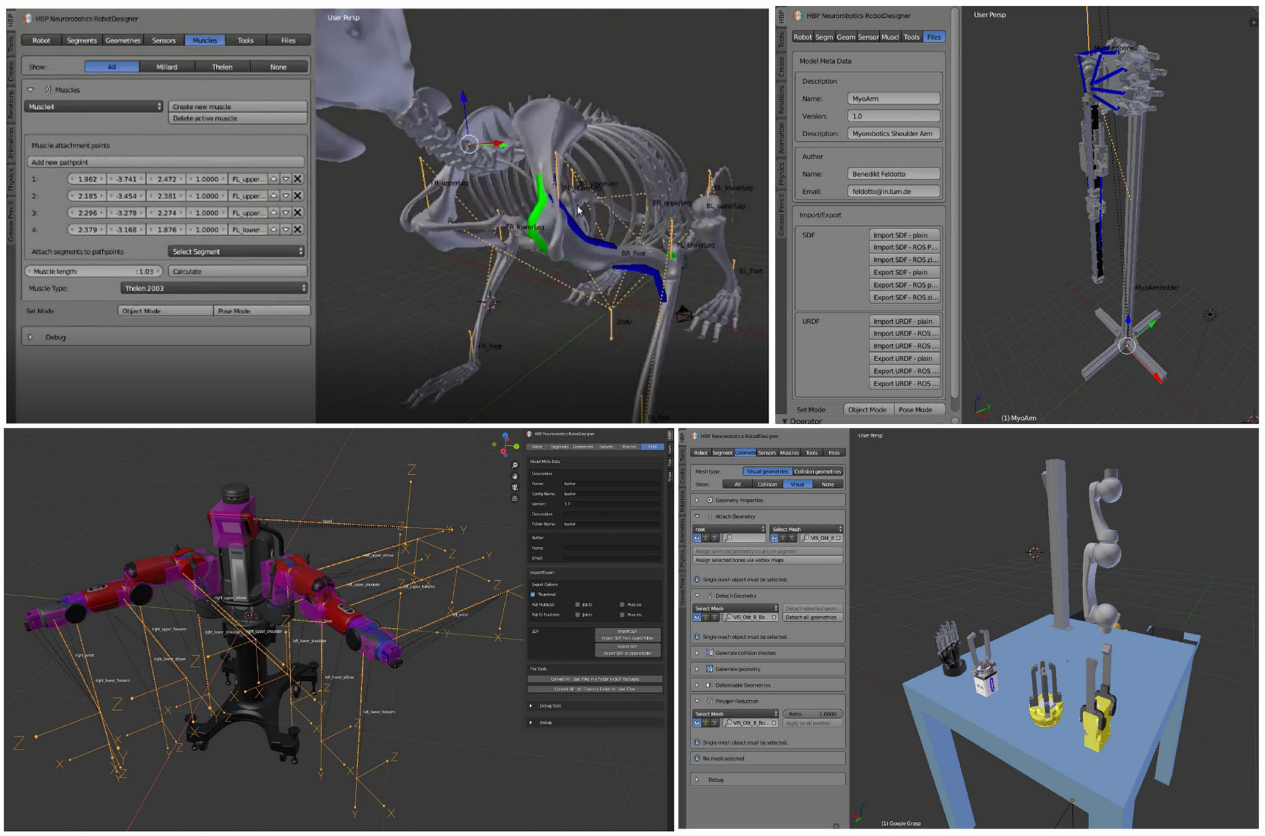
assignment to the body model can be done in a manner similar to basic collision geometries. Wrapping objects are Blender meshes tagged as wrapping object, they are color coded in the 3D view for visualization and graphical adaptation.

9. WORLDS

In the study of embodied intelligence, the agent's body is important, but only inasmuch it enables interactions with the environment. In particular, by closing the action-cognition-perception loop, an agent can proactively change the environment with actions mediated by the body, thereby perceiving new sensory states the nature of which essentially depends on the environment. We, therefore, implemented a dedicated user interface section that enables the user to assemble environments by adding body models that can be either actuated robots or static objects such as a walls, shelves and books. Finally, the environment can be customized adding lights and adapting parameters such as gravity. This parameterization is particularly interesting for simulation to reality transfer of neural network applications, as randomization of such settings may cover a multitude of possible real world states. A Robot Designer world is implemented as a Blender empty object that sets the reference frame. It is extended with various parameters for the environment (e.g., gravity, wind), general physics engine parameters (e.g., max_step_size, real_time_factor, real_time_update_rate, max_contacts) and simulation engine (e.g., ODE, SimBody, OpenSim) specific parameters for contacts and friction. The world object holds a list of referenced models as custom properties.

10. IMPORT AND EXPORT

Robot Designer models are particularly designed for, but not limited to, simulation in the Neurorobotics Platform; therefore, import/export functionalities are added accordingly. Models generated by the Robot Designer can be exported and imported in the SDFformat, linked geometries are exported as collada (.dae) mesh files. Technically, the exporter walks trough the armature from root to all branched links recursively and conglomerates the SDF specified parameters, both from the original Blender properties and from the custom Robot Designer properties that are assigned by the user. The SDFormat description does not itself contain an explicit graph description, but rather every link is defined with its parent and child. Therefore, importing a model from SDF first the link without parent or "world" as parent is found and then again recursively all child links are imported. As a result of this process, Blender objects are created in the 3D scene and the element specific properties are assigned to these objects. All geometries are exported in a dedicated mesh folder using the Blender Collada exporter and referenced in the SDF description. Additionally, where relevant, an OpenSim (.osim) file is generated holding the muscle description. For this purposes all muscle objects assigned to the given model are searched and exported as XML objects. Alongside every muscle, any referenced wrapping object is exported as well by adding its position, size and orientation parameters to the .osim description. Optionally, configurations for the ROS (tested with ROS Noetic) graphical user interface framework "RQt" (Thomas, Dirk and Scholz, Dorian and Blasdel, Aaron, 2016) can be exported. These include the configuration of plots for the rqt_multiplot package (Kaestner, Ralf, 2016),



1:	1.962	-3.741	2.472	1.0000	FL_upper...
2:	2.185	-3.454	2.381	1.0000	FL_upper...
3:	2.296	-3.278	2.274	1.0000	FL_upper...
4:	2.379	-3.168	1.876	1.0000	FL_lower...

Muscle length: 1.03

Name: MyoArm
Version: 1.0
Description: Myorobotics Shoulder Arm

Author
Name: Benedikt Feldotto
Email: feldotto@in.tum.de

FIGURE 5 | Example Robot Designer models: With the Robot Designer plugin, musculoskeletal models based on biological principles, or biomimetic robots exploiting biological characteristics, as well as classical robot models can be designed. From top left to bottom right: Musculoskeletal rodent model with skeleton and muscles, biomimetic robotic arm built from the modular Myorobotics toolkit with tendon actuators, humanoid robot Baxter, Schunk robotic arm with multiple grippers. In order to simplify the design process, visual cues are used and can be enabled optionally: orange lines indicate the kinematic tree, muscle colors indicate different muscle types, translucent purple boxes visualize the physics inertia. The user interface is kept consistent for Blender 2.7 and 2.8 (Baxter model).

for sensory data (joint position, velocity, effort and/or muscle length, lengthening speed, force) as XML description file, and the YAML file description of GUI sliders provided by the rqt_ez_publihser package (Ogura, Takashi, 2016) in order to send commands to joint controllers and muscle actuators interactively. An additional model configuration file is generated for metadata such as author name and model description, and a thumbnail image showing the robot after applying Blender's rendering is exported alongside the model files.

The overall package of exported files describing rigid body or musculoskeletal models can be used directly in simulations running in the Neurorobotics Platform. Musculoskeletal models can also be simulated with the standalone Gazebo version shipped with the NRP that supports the OpenSim physics engine, models without muscles in standard Gazebo with different physics engines. Additionally, parts of the exported models might be used in other simulators as well. The exported SDFormat description of rigid body models can be simulated in any other framework supporting SDF, the .osim muscle description file can be reused in the OpenSim simulator. Model simulation has been tested with SDFormat 6.0 (SDF protocol 1.6) in Gazebo 9 and OpenSimDocument version 30000 supported by OpenSim 3.2, respectively.

11. EXAMPLE MODELS

The design capabilities provided by the Robot Designer plugin allow users to create a variety of simulation models. In particular, different model types can be created that range from classical robots, musculoskeletal models, and biomimetic robots as a mix of both. **Figure 5** shows a selection of generated models that are generated or can be edited with our plugin: a musculoskeletal rodent model with several graphically defined muscles, a biomimetic robotic arm built with the Myorobotics toolkit and actuated with muscle-like tendon actuators, the humanoid robot "Baxter," a Schunk robotic arm with different types of grippers.

Various additional color-coded 3D visualizations can be enabled throughout the design process. For example, colors indicate different muscle types on the rodent model, while the Baxter robot is visualized with its inertia boxes in translucent purple. The various design tools are aligned with simulation capabilities in the Neurorobotics Platform, and therefore form

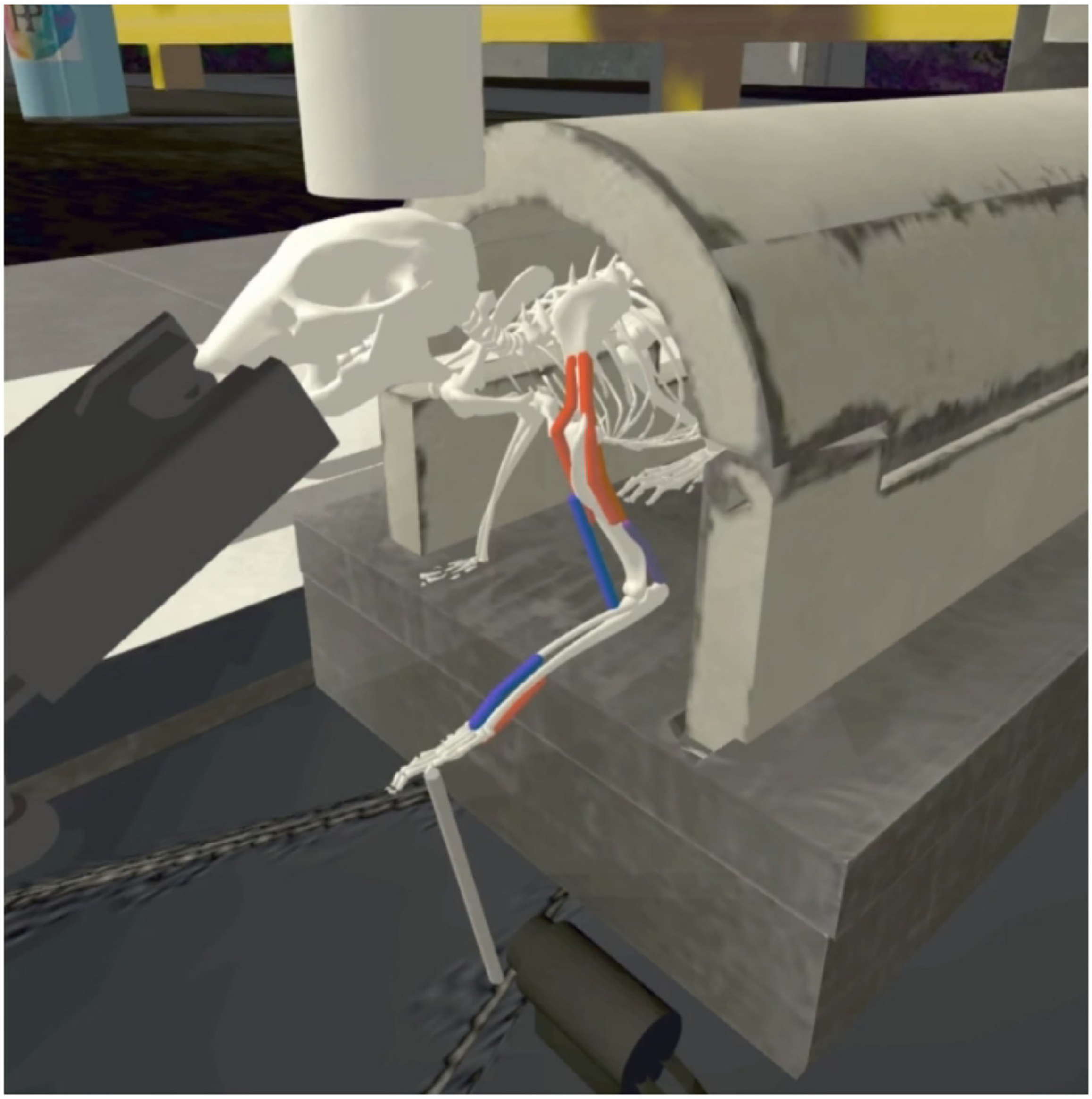


FIGURE 6 | Model simulation in the Neurorobotics Platform: The musculoskeletal rodent model and experiment platform setup were modeled or adapted and aggregated with the Robot Designer. After exporting the model, it was simulated in the Neurorobotics Platform, and individual muscles could be controlled independently *via* ROS topics (active: red, inactive: blue).

a strong asset for morphological design of agents for embodied simulation. Several models and environments available in the Neurorobotics Platform have been designed utilizing the Robot Designer plugin. As an example for the synergistic potential of both rigid body simulation fitted with muscles, in a recent paper (Mascaro et al., 2020) a rodent model has been set up using the Robot Designer for a stroke rehabilitation experiment. **Figure 6** shows the rodent model adapted with additional muscles, a lickometer and an enhanced joystick controller for a 3D manipulation experiment, that were added using the Robot Designer. First, the kinematic structure and physical properties were defined, then muscle path and attachment points were designed and fine-tuned using the graphical user interface.The image shows the final simulation in the Neurorobotics Platform.

12. CONCLUSION

In this article, we have described the features of the Robot Designer plugin integrated into the 3D modeling suite Blender, which enables users to create and parameterize robot and musculoskeletal simulation models. In this process, users can exploit the capabilities for mesh editing of Blender, as well as robot- and muscle-specific features provided by our plugin in a single framework. We introduced tools for kinematic and dynamics definition, as well as mesh adaptation and optimization for both visual appearance and physics simulation. The plugin described herein provides a user-friendly and graphic centric approach for simulation model design. Its tool suite enables building better models, and also helps users to faster and more easily adapt model parameters for studying the impact of morphology on motion learning. The support of both robotic and musculoskeletal models in the Neurorobotics Platform and the Robot Designer enables users to investigate novel designs and control strategies for biomimetic robots that combine characteristics from both classical rigid robots and muscle-like actuators. The capabilities of the plugin for model export and import were specifically designed to align with the Neurorobotics Platform. However, because it relies on community-standard model description and muscle specifications, the Robot Designer may also be used outside of the ecosystem of the Neurorobotics Platform, with Gazebo or OpenSim standalone, for example. The Robot Designer code is open source (GNU GPL license) and the project will be continued considering user contributions and requests.

To the best of our knowledge, we here demonstrate the first modeling suite that integrates both robot and musculoskelal modeling in a single design framework. The support of muscle descriptions is a key feature that distinguishes our approach from existing tools such as the Phobos robotic design plugin. The graphical user interface aims to support a straightforward design process of simulation models, a process which still is often done by manually changing description files.

The Robot Designer can play an important role in current research fields of embodied neuroscience and biomimetic robots and will, therefore, be enhanced driven on future research requests. In future improvements, we will add support for sensor import/export and additional functionalities such as modularization with a library of morphology parts that can enhance but also hasten body design. Besides the improvement of the Robot Designer itself, further developments will also provide an enhanced interface to the Neurorobotics Platform which will allow scripted adaption of morphologies throughout experimental epochs, and/or experiments that generate optimized morphologies based on evolutionary development.

The Robot Designer can play an important role in current research fields of embodied neuroscience and biomimetic robots and will therefore be enhanced driven on future research requests. In future improvements, we will add support for sensor import/export and additional functionalities such as modularization with a library of morphology parts that can enhance but also hasten body design.

Besides the improvement of the Robot Designer itself, further developments will also provide an enhanced interface to the Neurorobotics Platform which will allow scripted adaption of morphologies throughout experimental epochs, and/or experiments that generate optimized morphologies based on evolutionary development.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found at: GitHub, https://github.com/HBPNeurorobotics/BlenderRobotDesigner.

AUTHOR CONTRIBUTIONS

BF, FM, and AK conceptualized the Robot Designer extension for Neurorobotic modeling presented in this paper, building up on components of the predecessor RobotEditor, and wrote the paper. BF and FM orchestrated the implementation of the software project. BF is currently responsible for the Robot Designer code and user support, implemented the code for the new components, reviewed and integrated related code contributions, and build up most of the example models presented in this article. All authors contributed to the article and approved the submitted version.

FUNDING

The predecessor RobotEditor was funded by the European Commission under the Information Society Technologies of the seventh Framework Programme (FP7). This project has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements No. 720270 (Human Brain Project SGA1), No. 785907 (Human Brain Project SGA2), and 945539 (Human Brain Project SGA3).

ACKNOWLEDGMENTS

The authors would like to thank all code contributors including students at the Technical University of Munich, as well as the contributors to the predecessor RobotEditor at the Humanoids and Intelligence Systems Lab and at the Institute for Anthropomatics and Robotics of the Karlsruhe Institute of Technology.

REFERENCES

Albanese, U., Sharma, K., Feldotto, B., Akl, M., Weber, S., Mahmud, H., et al. (2020). HBP Neurorobotics Platform, Version 3.0. *Zenodo*. doi: 10.5281/zenodo.3763356

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). OpenAI Gym. *arXiv [Preprint]*. arXiv:1606.01540. Available online at: https://arxiv.org/pdf/1606.01540.pdf

Collins, J., Chand, S., Vanderkop, A., and Howard, D. (2021). A review of physics simulators for robotic applications. *IEEE Access* 9, 51416–51431. doi: 10.1109/ACCESS.2021.3068769

Community, B. O. (2018). *Blender - a 3D Modelling and Rendering Package*. Amsterdam: Blender Foundation, Stichting Blender Foundation.

Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., et al. (2007). OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE*

Trans. Biomed. Eng. 54, 1940–1950. doi: 10.1109/TBME.2007.901024

Euler, L. (1968). "Novi commentarii academiae scientiarum petropolitanae (1775) 1776; 20: 189-207," in *Reprint in: Leonhardi Euleri Opera Omnia, Formulae Generales Pro Translatione Quacunque Corporum Rigidorum. Series Secunda.* vol. 9, ed. C. Blanc (Basel: Orell Füssli Turici), 84–98.

Falotico, E., Vannucci, L., Ambrosano, A., Albanese, U., Ulbrich, S., Vasquez Tieck, J. C., et al. (2017). Connecting artificial brains to robots in a comprehensive simulation framework: the neurorobotics platform. *Front. Neurorobot.* 11, 2. doi: 10.3389/fnbot.2017.00002

Feldotto, B. (2017). *1.3 Robot Designer Demo*. https://www.youtube.com/watch?v=_ii0CVzVcsA

Fosselius, A. (2017). *FreeCAD RobotCreator Workbench*. https://github.com/maidenone/RobotCreator (accessed online at December 7, 2021).

Gewaltig, M.-O., and Diesmann, M. (2007). NEST (neural simulation tool). *Scholarpedia* 2, 1430. doi: 10.4249/scholarpedia.1430

Hartenberg, R., and Denavit, J. (1964). *Kinematic Synthesis of Linkages*. New York, NY: McGraw-Hill.

Kaestner, Ralf (2016). *rqt_multiplot ROS Noetic*. http://wiki.ros.org/rqt_multiplot

Knoll, A., Gewaltig, M.-O., Sanders, J., and Oberst, J. (2016). Neurorobotics: a strategic pillar of the human brain project. *Sci. Robot.* 25–35. Available online at: http://archive.www6.in.tum.de/www6/Main/Publications/knollNeuro2016.pdf

Koenig, N., and Howard, A. (2004). "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sendai), 2149–2154.

Leon B., Ulbrich, S., Diankov, R., Puche, G., Przybylski, M., Morales, A., et al. (2010). "OpenGRASP: a toolkit for robot grasping simulation," in *2nd International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)* (Darmstadt).

Marques, H. G., Maufroy, C., Lenz, A., Dalamagkidis, K., and Culha, U. (2013). "Myorobotics: A modular toolkit for legged locomotion research using musculoskeletal designs," IN *International Symposium on Adaptive Motion of Animals and Machines (AMAM) 6, 2013* Darmstadt.

Mascaro, A. L. A., Falotico, F., Petkoski, S., Pasquini, M., Vannucci, L., Tort-Colet, N., et al. (2020). Experimental and computational study on motor control and recovery after stroke: toward a constructive loop between experimental and virtual embodied neuroscience. *Front. Syst. Neurosci.* 14, 31. doi: 10.3389/fnsys.2020.00031

McGeer, T. (1990). Passive dynamic walking. *Int. J. Robot. Res.* 9, 62–82.

Müller, V. C., and Hoffmann, M. (2017). What is morphological computation? on how the body contributes to cognition and control. *Artif. Life* 23, 1–24. doi: 10.1162/ARTL_a_00219

Ogura, Takashi (2016). *rqt_ez_publisher, ROS Noetic*.http://wiki.ros.org/rqt_ez_publisher

Open Source Robotics Foundation (2020). *SDFormat*. www.sdformat.org

Pfeifer, R. (2000). On the role of morphology and materials in adaptive behavior. *From Anim. Anim.* 6, 23–32. doi: 10.7551/mitpress/3120.003.0004

Pfeifer, R., and Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. Cambridge: MIT Press.

Roboy development team (2021). *SDFusion*. https://github.com/Roboy/SDFusion (accessed online at: December 7, 2021).

Stanford Artificial Intelligence Laboratory et al. (2021). *Robotic Operating System (ROS)*. https://www.ros.org.

StephenBrawner (2021). *sw_urdf_exporter.* http://wiki.ros.org/sw_urdf_exporter (accessed online at: December 7, 2021).

Thomas, D., Scholz, D., and Blasdel, A. (2016). *RQT ROS Noetic*. http://wiki.ros.org/rqt

Vahrenkamp, N., Kröhnert, M., Ulbrich, S., Asfour, T., Metta, G., Dillmann, R., et al. (2012). "Simox: a robotics toolbox for simulation, motion and grasp planning," in *International Conference on Intelligent Autonomous Systems (IAS)* (Jeju-do), 585–594.

Vernon, D., Lowe, R., Thill, S., and Ziemke, T. (2015). Embodied cognition and circular causality: on the role of constitutive autonomy in the reciprocal coupling of perception and action. *Front. Psychol.* 6, 1–9. doi: 10.3389/fpsyg.2015.01660

von Szadkowski, K., and Reichel, S. (2020). Phobos: a tool for creating complex robot models. *J. Open Source Softw.* 5, 1326. doi: 10.21105/joss.01326

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

frontiers | Frontiers in Neurorobotics

REVIEW
published: 25 April 2022
doi: 10.3389/fnbot.2022.861825

Dexterous Manipulation for Multi-Fingered Robotic Hands With Reinforcement Learning: A Review

Chunmiao Yu[1] *and Peng Wang*[1,2,3,4*]

[1] *Institute of Automation, Chinese Academy of Sciences, Beijing, China,* [2] *School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China,* [3] *CAS Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, China,* [4] *Centre for Artificial Intelligence and Robotics, Hong Kong Institute of Science and Innovation, Chinese Academy of Sciences, Hong Kong, China*

With the increasing demand for the dexterity of robotic operation, dexterous manipulation of multi-fingered robotic hands with reinforcement learning is an interesting subject in the field of robotics research. Our purpose is to present a comprehensive review of the techniques for dexterous manipulation with multi-fingered robotic hands, such as the model-based approach without learning in early years, and the latest research and methodologies focused on the method based on reinforcement learning and its variations. This work attempts to summarize the evolution and the state of the art in this field and provide a summary of the current challenges and future directions in a way that allows future researchers to understand this field.

Keywords: dexterous manipulation, multi-fingered robotic hand, reinforcement learning, learn from demonstration, sim2real

OPEN ACCESS

Edited by:
Xin Luo,
Chongqing Institute of Green and Intelligent Technology (CAS), China

Reviewed by:
Bin Zhi Li,
Chongqing Institute of Green and Intelligent Technology (CAS), China
Gongfa Li,
Wuhan University of Science and Technology, China
Miao Li,
Swiss Federal Institute of Technology Lausanne, Switzerland

****Correspondence:***
Peng Wang
peng_wang@ia.ac.cn

Received: *25 January 2022*
Accepted: *22 March 2022*
Published: *25 April 2022*

Citation:
Yu C and Wang P (2022) Dexterous Manipulation for Multi-Fingered Robotic Hands With Reinforcement Learning: A Review.
Front. Neurorobot. 16:861825.
doi: 10.3389/fnbot.2022.861825

INTRODUCTION

Robotics has been a topic of interest for researchers for decades, and dexterous manipulation is one of the hottest these days. Although some simple tasks in the industrial environment have been solved, we also wish the robot can help us in some unstructured environments such as the domestic environment (e.g., helping blind people with daily routines) and some dangerous environments (e.g., nuclear decommissioning). Hence, the ability to operate with the dexterity of the robot is necessary. There are several definitions of dexterous manipulation problem, among which the one proposed by Bicchi (2000) is thorough and widely accepted: dexterous manipulation is the capability of changing the position and orientation of the manipulated object from a given reference configuration to a different one, arbitrarily chosen within the hand workspace.

In a structured environment where the shape of the objects is unaltered, the simple gripper is sufficient for simple tasks such as the pick-and-place task, and the gripper has more advantages in these tasks on account of its low price, easy control, and strong robustness. However, the dexterity of parallel claws is limited and they are not adapted to various objects and tasks. One solution is designing specific end-effectors for different objects and tasks. In a structured environment, this method is effective, but when facing a complex unstructured environment where one robot needs to deal with a lot of tasks and one robot needs to carry different end-effectors for different tasks, it is unpractical. Also, someone people argued that a dexterous arm with a simple gripper may be sufficient (Ma and Dollar, 2011). They pointed out that in some cases where the hand is for simple grasping and the arm is for manipulation, a dexterous arm with a simple gripper is sufficient and appropriate for many manipulation tasks. However, for some complex tasks such as in-hand manipulation, a simple gripper is not sufficient and a multi-fingered dexterous hand is, therefore, necessary. **Figure 1** shows the typical tasks of dexterous manipulation with multi-fingered robotic hands including pouring (Qin et al., 2021), dexterous grasping (Li et al., 2014), object relocation (Rajeswaran et al., 2018), and so on, which are difficult or impossible to be accomplished by simple manipulators.

A dexterous hand can greatly improve dexterity and increase the workspace of the system. Additionally, the application of the dexterous hand can reduce the energy required for the task due to the lower feedback gains required as opposed to a full arm.

When mentioning a dexterous manipulator, the first thing that comes to mind is the human hand. Even some philosophers deem that it is the dexterity of the human hand that leads to human intelligence. Therefore, it is no surprise that most robot hands designed for dexterous manipulation are similar to the human hand in both shape and structure. The past several decades have seen the emergence of many dexterous multi-fingered hands. In 1984, the Center for Engineering Design at the University of Utah, and the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology designed the UTAH/MIT hand with three fingers and a thumb aiming at machine dexterity (Jacobsen et al., 1986), and later HIT developed the DLR/HIT Hand II (Liu et al., 2008). Also, there are some

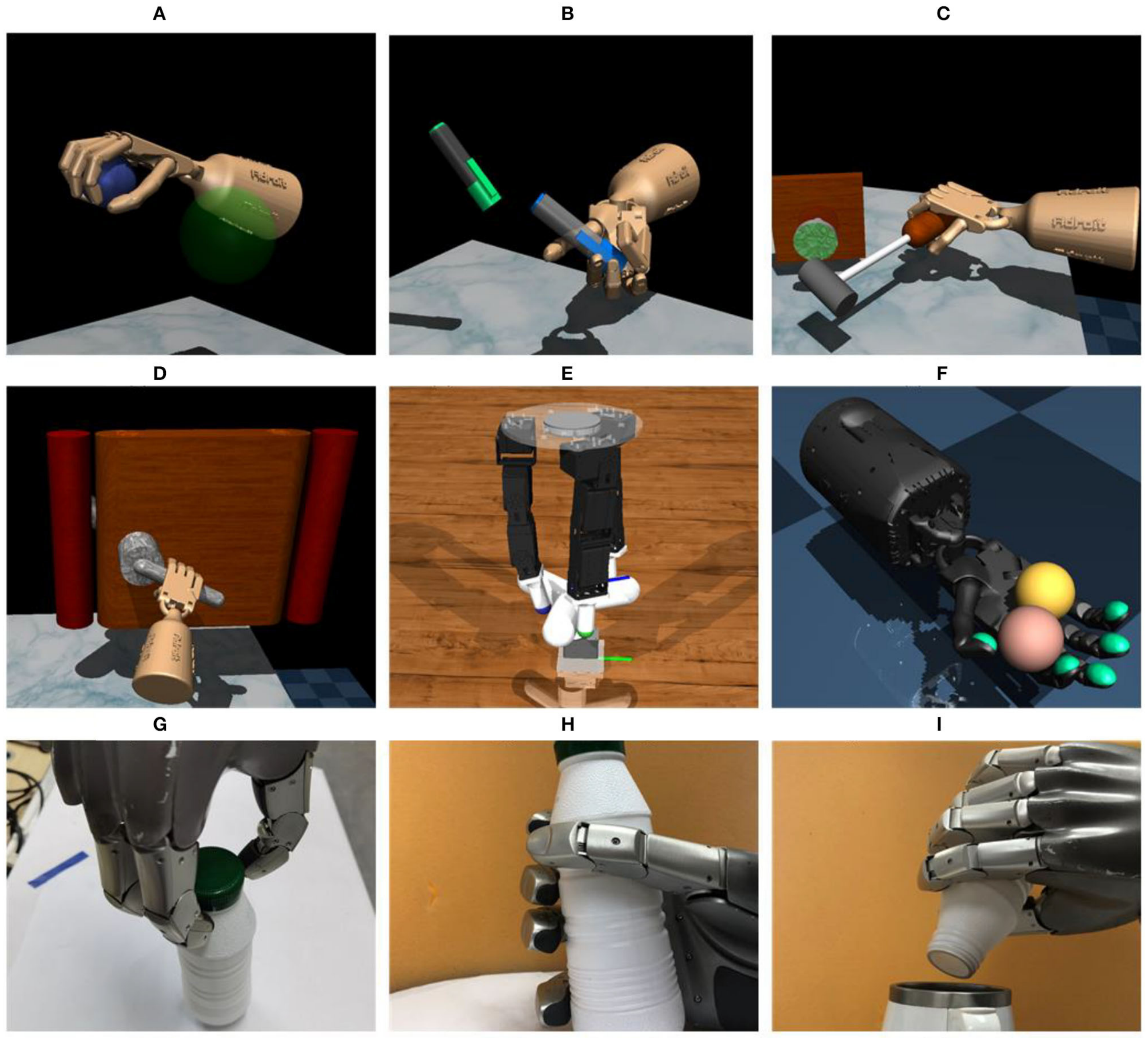
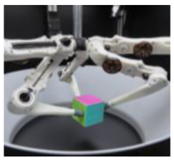


FIGURE 1 | Typical tasks of dexterous manipulation with a multi-fingered hand. **(A)** Relocation, **(B)** Reorientation & Relocation, **(C)** Tool use, **(D)** Door opening, **(E)** Valve turning, **(F)** In-hand manipulation, **(G)** Screwing, **(H)** Dexterous manipulation, and **(I)** Pouring.

TABLE 1 | Typical dexterous hands.

Name	TriFinger (Wüthrich et al., 2021)	Dclaw (Zhu H. et al., 2018)	Utah/MIT (Jacobsen et al., 1986)	Allegro (Allegro hand)	Shadow (Shadow Hand, 2005)	DLR/HIT II (Liu et al., 2008)
Picture						
Fingers	3	3	4	4	5	5
DoF	9	9	16	16	24	15

commercial products such as the Shadow Hand (2005) and SimLab (Allegro hand). Apart from the dexterous humanoid robotic hands, some simpler robotic manipulators with fewer fingers and a lower dimension of freedom (DoF) are also designed for better robustness and lower price (Zhu H. et al., 2018; Wüthrich et al., 2021). Some common multi-fingered robotic hands and some important parameters are shown in **Table 1**. However, up to now, the dexterity of the human hand is still unparalleled and it is scarcely possible to emulate the level of its functionality.

The current applications of robotic hands in the factories still use traditional engineering and analysis techniques. Typically, some robots with simple end-effectors are widely used in the manufacturing industry for packaging and palletizing. Similarly, agricultural robotic hands with several end-effectors and painting robotics are good examples for the application of robotic hands in the structured industry environment. Although the dexterous manipulation problem has been studied extensively, the application of the learning-base methods in this review still remain at the laboratory level, which is not sufficient for unstructured environment such as businesses and homes.

Although the mechanical design of smart manipulators has improved greatly, the actual dexterity of the robotic hands is far inferior to that of the human hand. On the one hand, lots of sensors and actuators of the human hand makes it almost impossible to design a robotic hand which is similar to the human hand (Billard, 2019), and on the other hand, the control of the robotic hand to realize dexterous manipulation is still an urgent problem to solve. Before 2000, the approach was based on the kinematics and dynamics of manipulating an object with the fingertips dominating the area. This approach requires the complete information of the manipulator kinematics, dynamics, interaction forces, high-fidelity tactile, and/or joint position sensors available on-board the robot. However, the accurate model of the environment and the object is not or partly available in the real world. Moreover, even though the information is available, the algorithm must change as the object or the manipulator changes. Hence, in the real world, the model-based approach has certain limitations.

Recently, the power of artificial intelligence has attracted the attention of many researchers. Deep learning has even reached a level that exceeds that of humans in certain fields, such as computer vision, so the robot can extract generalized features autonomously (LeCun et al., 2015; Duan et al., 2021; Wei et al., 2021, 2022; Li et al., 2022). Deep learning is better at classification and prediction problems and so on. But the application of deep learning is still short of the entire system model. In contrast, reinforcement learning (RL) is more suitable for dealing with the sequential decision problem. Therefore, the combination of deep learning and reinforcement learning called deep reinforcement learning is proposed to realize more complicated problems involving perception and decision making. Dexterous manipulation is a typical decision-making problem, so deep reinforcement learning, as it were, dominated the area in recent years. However, the application of deep reinforcement learning to dexterous manipulation has some disadvantages. First, the sparse reward makes the training hard, and for complex tasks, it is time-consuming and the requirement of computing power is high. Furthermore, deep reinforcement learning requires many samples obtained by trial and error, which are nearly unavailable in a robotic system. To solve this problem, besides the improvement of the RL algorithm, usually two solutions are considered: learning from demonstration and transferring the policy learned in simulation to reality. These two approaches will greatly enhance the efficiency of the algorithm.

There are already several works reviewing the robot manipulation domain (Billard, 2019; Cui and Trinkle, 2021), reinforcement learning for the robot (Hua et al., 2021; Zhang and Mo, 2021), and dexterous manipulation only (Prattichizzo et al., 2020). However, as far as we know, a survey focusing on dexterous manipulation with multi-fingered robotic hands with reinforcement learning has never been presented before. Here, we present a review of this domain including the method based on dynamic analysis in the earlier years and the reinforcement learning-based method in recent years. Although the method based on reinforcement learning is the core of this paper, we think the method based on dynamic analysis is necessary for readers to understand the dexterous manipulation problem.

The main contribution of this paper is presenting a state-of-the-art review focused on the dexterous manipulation problem of multi-fingered robotic hands with reinforcement learning. The paper first reviews the model-based approach without learning including the basic modeling, planning, and control. Further, the methods based on deep reinforcement learning,

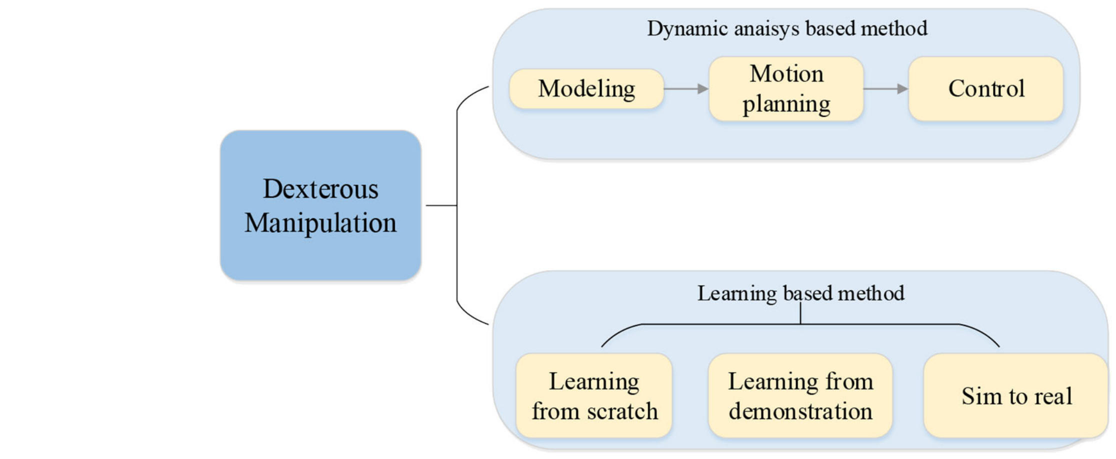


FIGURE 2 | Overall presentation of this work.

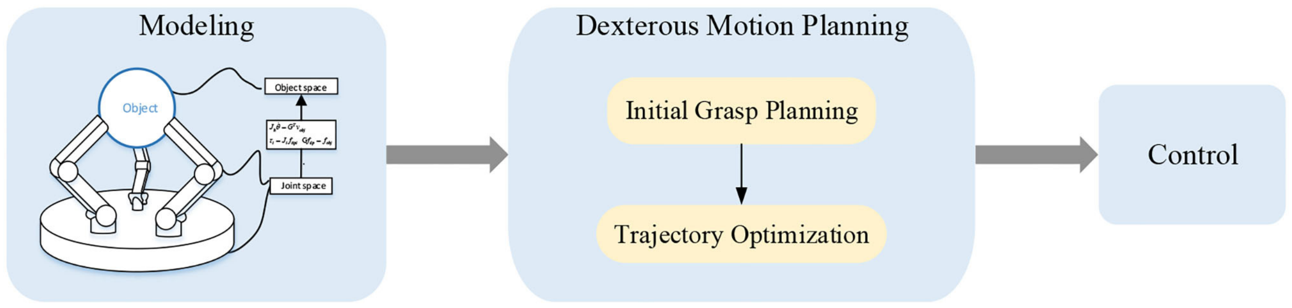

Modeling
Object
Object space
$J_h\dot{\theta} - G^T v_{obj}$
$\tau_i - J_i f_{tipi}$ $Gf_{tip} - f_{obj}$
Joint space
Dexterous Motion Planning
Initial Grasp Planning
Trajectory Optimization
Control

FIGURE 3 | Method based on an accurate model of multi-fingered hand and object.

reinforcement learning from demonstration, and transfer learning from simulation to reality are summarized and analyzed thoroughly. Finally, challenges and future research directions are proposed. The main topics discussed in this article are shown in **Figure 2**.

The rest of this article is organized as follows. After this introductory section, in Section Dexterous Manipulation for Multi-Fingered Robotic Hand Based on Modeling, Planning, and Control, we introduce the basic theory of dexterous manipulation including the model of the multi-fingered robotic hands and the object and the model-based approach for dexterous manipulation. Section Dexterous Manipulation for Multi-Fingered Robotic Hands With Reinforcement Learning focuses on the dexterous manipulation with reinforcement learning, including the application of reinforcement learning, the combination of reinforcement learning, and learning from demonstration and deploying the learned policy in simulation to the real world. At the same time, we also discuss the characteristics of the approaches mentioned in this paper. Section Challenges and Future Research Directions describes the current limiting factors in manipulation and look forward to the further development of dexterous manipulation.

DEXTEROUS MANIPULATION FOR MULTI-FINGERED ROBOTIC HAND BASED ON MODELING, PLANNING, AND CONTROL

The dexterous problem can be described as determining the contact points and the forces/torques that should be exerted upon the object and planning a trajectory to control the end-effector to accomplish a specific task. In this section, we will introduce the basic theories of dexterous manipulation including the models of contacts, positions, forces, and velocities; motion planning and the control framework for dexterous manipulation. The progress of the model-based approach including modeling, dexterous motion planning, and control are depicted in **Figure 3**.

Modeling of Multi-Fingered Robotic Hands and Objects

Usually, an object-centered point of view is adopted for describing the dexterous manipulation problem. The formulations are in terms of the object to be manipulated, how it should behave, and what forces should be exerted upon it. Therefore, the relationship of the desired forces/torques on the object and the required contact forces and the relationship of the required contact forces and the joint torques is required. Typically, the model of contact between the object and the fingertip can be seen as point-contact and the model of the robotic hand can be seen as a set of kinematic chains consisting of links connected by joints. The most popular method for formulating the forward kinematics of robots is the D-H method. Specifically, four D-H parameters are used for the transformation between two co-ordinate systems. More details about the D-H convention and the point-contact model can be seen in (Spong et al., 2006) and (Okamura et al., 2000), respectively.

In addition to maintaining the contacts during the manipulation process, rolling and sliding may sometimes occur during manipulation. Although sliding in some tasks is not allowed, the sliding mode is necessary when exploring an unknown object or changing the pose of a grasp to maintain control of the object. More details about rolling and sliding can be seen in Montana (1988) and Kao and Cutkosky (1992), respectively. However, sliding is rarely considered in the early years due to lack of reliable tactile sensors to keep track of the contact locations on the fingertips and indicate the onset of slip.

Dexterous Motion Planning of Multi-Fingered Robotic Hand

Typically, the dexterous manipulation problem can be divided into two parts, namely initial grasp planning and trajectory optimization which will be discussed, respectively, in this subsection.

Grasp Planning of Multi-Fingered Robotic Hands

To deal with dexterous manipulation, the first thing to be considered is stable grasping. Grasping generally consists of two phases: a planning phase and a holding phase. In the planning phase, the finger contact point locations are decided and the object is grasped stably in the holding space. Two important problems are considered for the two phases accordingly: the selection of feasible locations of contact and optimal contact forces.

Selection of Feasible Locations of Contact

Two important concepts describing the stability of a given grasp are force-closure and form-closure. We refer the readers to Bicchi (1995) for more details about force-closure and form-closure. However, force-closure is only the bottom-most condition to satisfy and not enough for a stable and desired grasp. Furthermore, in a specific task, there would be many configurations that achieve force-closure, so the problem that which one should be adopted is very important. Being on the safer side, an intuitive measurement is to apply less force on the object, resulting in a better grasp effect. The first one who proposed this idea was Kirkpatrick et al. (1992), and Ferrari and Canny (1992) improved it later. Similarly, for different consideration factors, a few metrics were proposed, such as task-oriented metrics (Hsu et al., 1988), eigenvalue decomposition-based metric (Bruyninckx et al., 1998), and metrics considering different issues (Lin and Burdick, 1999; Lin et al., 2000; Roa and Suárez, 2009). However, getting optimal contact locations through appropriate metrics and optimization methods is difficult due to that the quality measure is typically a non-convex (and non-linear) function. Besides the optimization approach, some researchers used a knowledge-based approach (Cutkosky, 1989; Stansfield, 1991) to get a suitable grasp.

Selection of Optimal Contact Forces

To generate a great grasp, we should plan not only the locations of the contact points but also the force exerted to the object on the contact. In early works, the friction constraint was linearized and the coefficient of friction was estimated conservatively to avoid instability and considering the problem as a non-linear programming problem (Nakamura et al., 1989; Nahon and Angeles, 1991; Al-Gallaf and Warwick, 1995). However, such methods are offline, and considering the problem in a non-linear context was also proposed for online implementation (Buss et al., 1996). These computed forces are then used in the low-level force servo mechanism to produce a desired force behavior in the object.

Trajectory Optimization for Dexterous Manipulation With a Multi-Fingered Robotic Hand

For relatively simple tasks, the contact points remain the same during the manipulation, so after getting the desired grasp configuration and contact forces, the task can be achieved by controlling the robot arm. However, for more complex tasks such as in-hand manipulation, one grasp is not sufficient. Therefore, a trajectory of grasps which links the initial grasp and the desired grasp is required.

The methods proposed in the dexterous manipulation problem are typically derived from the legged locomotion problem. However, the methods used in the legged locomotion are not suitable for hand movement control due to the high dimensions of the search space. A representative work proposed by Mordatch et al. (2012a) is an extension of contact-invariant optimization (CIO) (Mordatch et al., 2012b) which was used for character animation originally. However, the CIO is an offline method and time-consuming. In practice, online planning (or Model-Predictive Control) is more desirable (Kumar et al., 2014), where a trajectory of the control signal is optimized and the joint space trajectories are obtained through inverse kinematic (IK). For solving (Sundaralingam and Hermans, 2017) the in-grasp manipulation problem more directly, get a joint space trajectory without the process of IK. However, this approach requires maintaining the contacts, which is only a part of the whole dexterous manipulation process. With this in mind, Sundaralingam and Hermans (2018) presented a planner for reorientation of the object through finger gaiting and in-grasp manipulation alternately. Similarly, Chen C. et al. (2021a) proposed TrajectoTree, a method based on contact-implicit trajectory optimization (CITO). Unlike the optimization method, the concept of motion primitives is also accepted widely (Chen C. et al., 2021b; Yoneda et al., 2021). The phase of motion planning is the core of dexterous manipulation. However, only under certain assumptions can these approaches work, such as assuming that the shape and mass of the object are known and the contacts remain during the manipulation process

(Sundaralingam and Hermans, 2017). Also, some approaches can only be applied to planar objects (Chen C. et al., 2021a). At the same time, most of these methods are only tested in simulation. From what has been discussed above, the approaches based on trajectory optimization have many limitations for achieving dexterous manipulation with multi-fingered robotic hands in the real world.

The Control of Multi-Fingered Robotic Hand for Dexterous Manipulation

The control of multi-fingered robotic hands for dexterous manipulation can typically be divided into three levels. The high-level control includes grasp planning and motion planning which have been discussed thoroughly. The middle-level control, which is relatively unpopular compared to the other two levels, includes event detections and phases transitions. Hence, only a few researchers focus on this problem (Johansson and Westling, 1991; Eberman and Salisbury, 1994; Hyde et al., 1997; Hyde and Cutkosky, 1998).

The low-level control is a primary part of the dexterous manipulation problem and has received a lot of attention. Trajectory tracking in free space and precise force control in constrained space should be both taken into consideration. During tracking in free space, position control is enough because the robot hand does not make contact with the object at this stage. During the contact stage, position control and force control are both important for precise force. Taking both position control and force control into account, several control algorithms were put forward such as simple hybrid position/force control which is widely used (Raibert and Craig, 1981; Xiao et al., 2000), impedance control (Hogan, 1984), and the combination of hybrid position/force control and impedance control (Anderson and Spong, 1988). The impedance control can solve the problem of discontinuity by the change of the control mode, so it has attracted much attention of researchers (Goldenberg, 1988; Kelly and Carelli, 1988; Kelly et al., 1989). The combination can furthermore be considered as the distinction between force-controlled subspaces and position-controlled subspaces.

DEXTEROUS MANIPULATION FOR MULTI-FINGERED ROBOTIC HANDS WITH REINFORCEMENT LEARNING

Given that the complete model of the objects and robotic hand is difficult to obtain in an unstructured environment and programming robots require a high degree of expertise, the methods mentioned above are not sufficient for a more complicated environment and tasks. The development of machine learning, especially reinforcement learning, provides new solutions to the problem of dexterous manipulation with multi-fingered robotic hands. The whole progress of solving dexterous manipulation with reinforcement learning is shown in **Figure 4**. In this section, we will discuss dexterous manipulation with reinforcement learning and its variations.

Reinforcement Learning

The reinforcement learning problem is a kind of machine learning algorithm which learns mapping environment state to action and obtaining the maximum cumulative reward in the process of interaction with the environment. Q-learning is a traditional solution to the problem, however, it is not sufficient for more complicated problems today due to the high cost of solving the q-valued function with lots of states and actions. The combination of deep learning and reinforcement learning called deep reinforcement learning (DRL) was proposed for more complicated problems and it dominates the area now.

The method can be divided into the model-based method and model-free method, the difference between the two is whether a predictive model is used. The earliest model-based algorithm is Dyna (Sutton, 1990), where the model is learned by data from the real world and both the data from the real world and the learned model are used in the training process. There are some other model-based algorithms such as PILCO (Deisenroth and Rasmussen, 2011), M-PGPE (Mori et al., 2013), PEGASUS (Ng and Jordan, 2013), GPS (Levine and Abbeel, 2014), VPN (Oh et al., 2017), MVE (Feinberg et al., 2018), STEVE (Buckman et al., 2019), and MBPO (Janner et al., 2019). On the contrary, in the model-free method, the agent learns the strategy directly by interacting with the environment. The comparison between the model-based method and model-free method can be seen in **Table 2**. According to the characteristic of the model-based method and model-free method, the selection between the model-based method and model-free method is a crucial problem and should be taken into account.

Reinforcement learning also can be divided into three types according to the variables iterated in the learning process: value-based method, policy-based method, and actor-critic method. In the value-based method, the value function is learned and the policy is determined by a greedy strategy or a strategy. Deep Q-learning (DQN) (Mnih et al., 2015) and its variations (van Hasselt et al., 2015; Schaul et al., 2016; Wang et al., 2016) are typical model-free value-based method. Although DQN and its variants have achieved excellent performance in discrete action space problems such as video games, and even defeated human players by overwhelming advantage in some games, they cannot cope with the continuous action space problems that exist in many actual production and life such as dexterous manipulation.

Different from the value-based approach, the policy is straightly optimized in policy-based algorithms. REINFORCE (Williams, 1992) is a monumental algorithm which provides the state transition model-independent algorithm theoretically and becomes the starting point of many algorithm improvements. It plays a pioneering role in the algorithm system of policy gradient series represented by TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017). However, although TRPO and PPO algorithms have excellent hyperparameter performance and have gained attention in academic research as typical on-policy algorithms, many samples under the current policy need to be sampled for training and to ensure algorithm convergence each time the policy is updated. Therefore, the

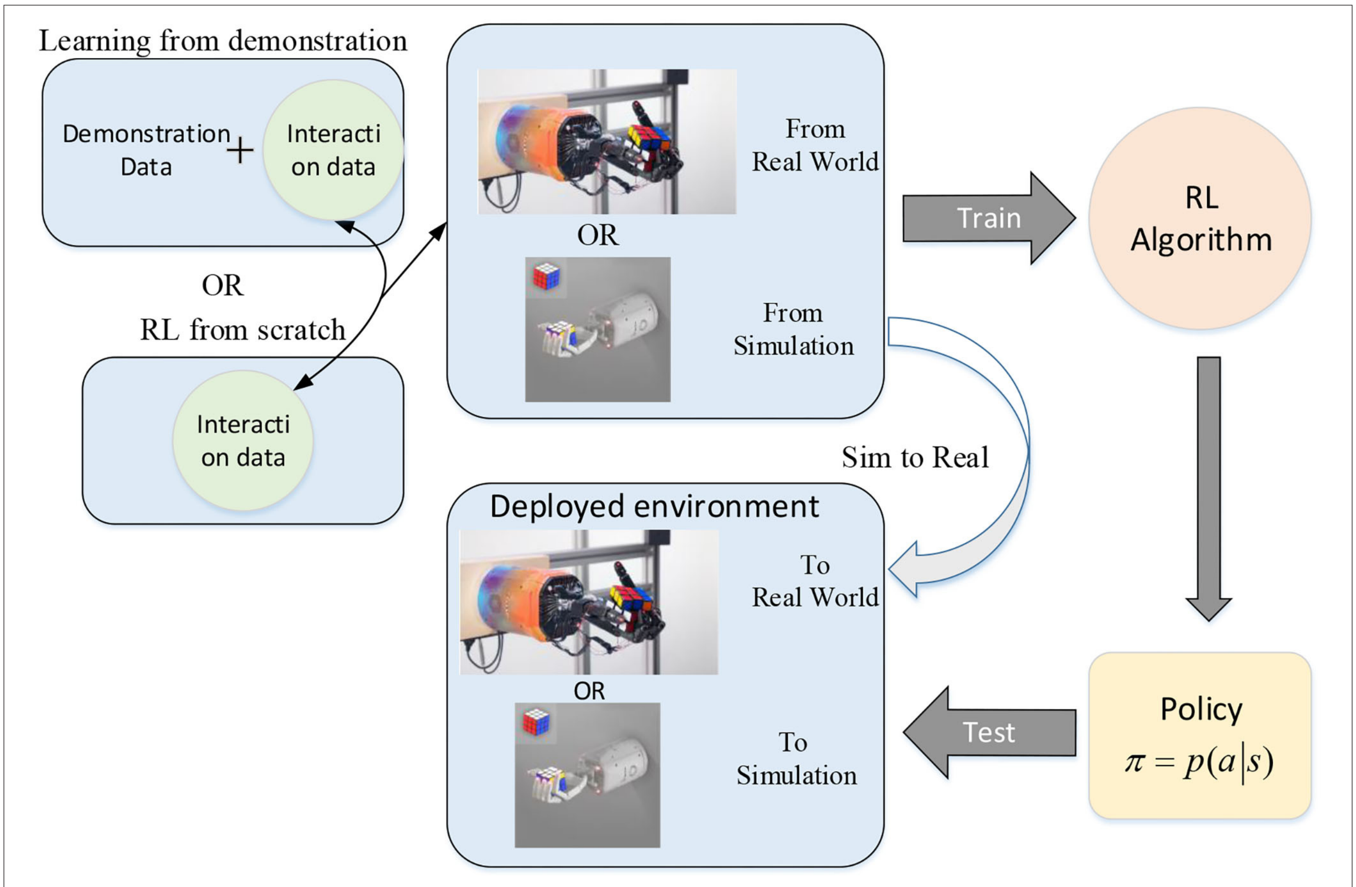


FIGURE 4 | Dexterous manipulation with a multi-fingered hand through reinforcement learning (part of this picture comes from [89]).

algorithms have low sampling efficiency and need a large amount of computational force to support, which greatly limits the popularization of the algorithms in the application field. A survey of the classification and corresponding comparison between the advantages and disadvantages of RL methods is shown in **Table 2**. Furthermore, a more detailed comparison between typical value-based algorithms and policy-based algorithms can be seen in **Table 3**.

As we listed in **Table 2**, an important problem of the policy-based method is high variance and the combination of the value-based method and a policy-based method called the actor-critic method can solve this problem to some extent. The state-of-the-art algorithms at present are all under the actor-critic framework. The typical RL algorithms under the actor-critic framework are summarized in **Table 4**.

The actor-critic algorithm is mostly off-policy and can solve the problem of sampling efficiency through experience replay. However, the coupling of the policy update and value evaluation results in the lack of stability of the algorithm, especially the sensitivity to hyperparameters. In the actor-critic algorithm, it is very difficult to adjust parameters, and the algorithm is also difficult to reproduce. When it is promoted to the application field, the robustness of the algorithm is also one of the most concerning core issues. Commonly, the data of reinforcement learning are often incomplete, so we refer the readers to the following literature (Shang et al., 2019; Luo et al., 2020; Wu D. et al., 2020; Wu et al., 2020; Liu et al., 2021) for more details.

TABLE 2 | Classification and corresponding advantages and disadvantages of the RL methods.

Classification	Advantages	Disadvantages
Value-based RL	1. Easy to implement 2. High sample utilization	1. Poor performance in tasks of discontinuous and large state space 2. High bias
Policy-based RL	1. Easier to converge 2. More directly	1. Easy to converge to local optimum 2. High variance
Model-based RL	1. More data efficient 2. Faster convergence	1. Model accuracy has a big impact on learning tasks
Model-free RL	1. Easier to implement 2. No need of prior knowledge	1. Demanding much data 3. High risk of damage

TABLE 3 | Comparison between typical value-based algorithms and policy-based algorithms.

Algorithm	Main characteristic	Value-based/ policy-based	Limitations
DQN (Mnih et al., 2015)	Approximating the optimal Q-value function with a deep convolutional neural network. Target-network and Experience replay	Value-based	Only capable of handling discrete and low-dimensional action spaces
Double DQN (van Hasselt et al., 2015)	Two networks are used for dealing with the overestimation problem of DQN	Value-based	
DQN with prioritized experience replay (Schaul et al., 2016)	Experience replay with priority is used to increase the learning utilization rate of samples and increase exploration	Value-based	
Dueling DQN (Wang et al., 2016)	V(s)+A(s, a) is used to replace Q(s, a) to alleviate the overestimation problem of DQN	Value-based	
REINFORCE (Williams, 1992)	The starting point of policy gradient algorithms	Policy-based	Low efficiency and high variance
TRPO (Schulman et al., 2015)	Finding the right step size to stably improve the policy	Policy-based	
PPO (Schulman et al., 2017)	An advanced version of TRPO which is easier to implement	Policy-based	

TABLE 4 | Summary of typical algorithms under the actor-critic framework.

Method	Main characteristic	Off-policy/ On-policy
A3C (Mnih et al., 2016)	Adopting asynchronous training framework	On-policy
DDPG (Lillicrap et al., 2015)	Able to deal with continuous space of action issues	Off-policy
TD3 (Fujimoto et al., 2018)	An advanced version of DDPG solving the problem of overestimation in actor-critic and addressing variance	Off-policy
SAC (Haarnoja et al., 2018)	Adopting Maximum Entropy Model to improve the robustness of the algorithm and speed up training	Off-policy

Dexterous Manipulation With Multi-Fingered Robotic Hands Using RL From Scratch

The success in various complex tasks such as reorienting an object (Open et al., 2019), tool use (Rajeswaran et al., 2018), and playing the piano (Xu et al., 2021) has shown the power of reinforcement learning for dexterous manipulation. For dealing with the dexterous manipulation problem under the framework of RL, the problem is usually modeled as a Markov decision process (MDP), where the states can be the combination of internal states and external states, and the action is typically the motor commands. In a simulation, the states are available, however the needed elements for states cannot be obtained directly. Under that condition, visual sensors and tactile sensors are usually used for inferring the state or using the raw sensor data as the state (Katyal et al., 2016). The easiest way to think of is to train the agent from scratch. The basic process of learning dexterous manipulation by RL from scratch is depicted in **Figure 5**.

Although learning-based methods are appealing to roboticists for dealing with the dexterous manipulation problem, the need for large amounts of data has always been a major obstacle to the development of robotics. Hence, most researchers focused on enhancing the sample efficiency but from various angles. Some of the researchers focus on the algorithm itself and test only in simulation (Popov et al., 2017; Haarnoja et al., 2019; Omer et al., 2021). Popov et al. (2017) decouples the update from the frequency of interaction and trades off between the exploration and the exploitation by defining certain starting states and shaping reward effortfully. Haarnoja et al. (2019) improved the SAC for accelerating training and improving stability. Omer et al. (2021) present MPC-SAC combining the Model-Predictive Control (MPC) which is an offline learning method with online planning, which can be seen as a model-based RL method. Similarly, model-based methods are also adopted in (Kumar et al., 2016; Nagabandi et al., 2020). Different approximators such as time-varying linear-Gaussian (Kumar et al., 2016) and deep neural network (Nagabandi et al., 2020) are used, respectively. Moreover, the combination of local trajectory optimization and RL is also attractive (Lowrey et al., 2019; Charlesworth and Montana, 2021). Fakoor et al. (2020) centered around the instability problem in RL and reduced the complexity in the famous state-of-the-art RL algorithms. Some researchers also pay attention to the problem of sparse reward which is a common hindrance in RL causing sample inefficiency. To this end, HER is a widely used algorithm which learns from failures and can be combined with any RL algorithm. Li S. et al. (2019) just incorporate HER in the hierarchical RL framework to achieve the complex Rubik's cube task. The introduction of HER in RL can also be seen in (He et al., 2020; Huang et al., 2021).

Besides the problem of sample inefficiency, generalization is another major obstacle yet to be bordered. As a rule, multi-task RL is a popular concept to the researchers in autonomous robots (Hausman et al., 2018; He and Ciocarlie, 2021; Huang et al., 2021). Considering the inefficient exploration caused by the high DoF of the dexterous robotic hand, which means the high dimension of action space, He and Ciocarlie (2021) proposed a lower-dimensional synergy space and multi-task policy. In contrast to exploring in the raw action space with high dimension, exploring in the synergy space can improve the efficiency in exploring new environments or learning new tasks. Similarly, Hausman et al. (2018) presented embedding space

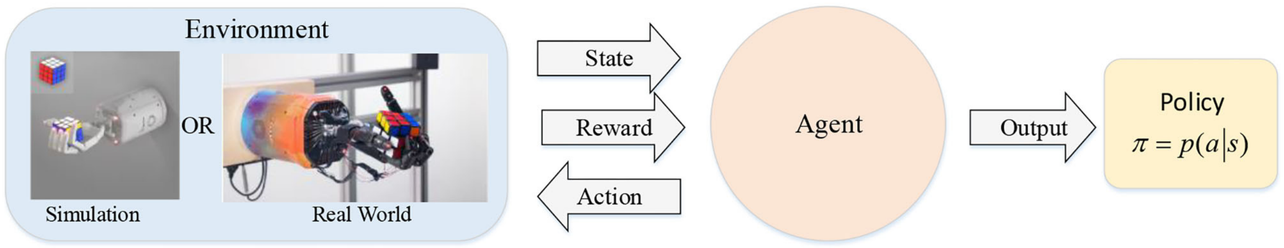


FIGURE 5 | Basic process of learning dexterous manipulation by RL from scratch (part of this picture comes from Open et al., 2019).

to the same end. What is different is that Huang et al. (2021) focused on one task on various objects other than different tasks. With the help of a well-designed object representation and multi-task framework, the manipulation of 70 different objects can be realized by one policy model achieving similar or better results than single-task oracles. The success of this work is a big step toward making robotic hands intelligent.

The previously mentioned works were only tested in a simulation where data were easy to get, however, a great performance in simulation cannot guarantee the performance. Furthermore, the elements required for representing the state are not available in the real world, so sensors are necessary for representing the state. As a rule, the visual sensor is the main consideration. For instance, Haarnoja et al. (2019) adopted the raw image as a representation of the state. Experiences implicate that the introduction of tactile information can effectively improve the sample efficiency for training and the performance in dexterous manipulation tasks (Melnik et al., 2019). van Hoof et al. (2015) used the tactile sensor data and introduced the non-parametric relative entropy policy (NPREPS), which is well-suited to the sensor data. Falco et al. (2018) used the visual sensor and tactile sensor together. The visual sensor is used for representing the state in the RL process and the tactile sensor acts as feedback in a low-level reactive control aiming at avoiding slipping. Also, training on a real robotic hand usually costs time and requires human intervention. To alleviate the problem, Gupta et al. (2021) proposed a reset-free reinforcement learning algorithm. They pointed out that the learning of multi-task and sequencing them appropriately can solve the problem naturally. The algorithm achieved great performance both in simulation and the real world.

All the details of the above works are listed in **Table 5**, including the specific method, the environment (e.g., simulation or real world or from simulation to real world), the manipulator, sensors utilized, and the tasks.

Dexterous Manipulation With Multi-Fingered Robotic Hands Using Reinforcement Learning From Demonstration

Apart from improvement on the RL algorithm, some researchers were inspired by the way learners paid attention to learning from demonstrations, which is also called imitation learning. An intuitive idea is following the expert demonstrations in a supervised way, namely behavior cloning. However, the policy depends on the expert data too much in this way. Another common method in imitation learning is inverse reinforcement learning where the reward function is learned. The introduction of demonstration data in reinforcement learning is an effective approach for enhancing the sample efficiency and the generalization performance in behavior cloning only.

The sources of demonstrations can be kinesthetic teaching, teleoperation (Zahlner S. et al., (n.d.); Handa et al., 2019; Li T. et al., 2019; Li et al., 2020), raw video, and so on. The problem of learning from demonstration has been studied a lot in recent years and a comprehensive survey can be seen in Ramírez et al. (2021). Ramírez et al. (2021) divided the use of the demonstrations into two types of knowledge: prior knowledge and online knowledge. In the case of the former, the demonstration data were stored before the RL process and acted as source of knowledge such as being added to the reward function for bringing the policy closer to the demonstration. In the case of the latter, the demonstrations are used occasionally to provide a trajectory. The process of the two types of combination can are depicted in **Figure 6**.

Here we follow the same sort of classification and go further into the application in dexterous manipulation with multi-fingered robotic hands. In the first class, the demonstrations can be utilized in various ways. For instance, a kinesthetic demonstration is adopted as the desired position trajectory as prior knowledge to get an initial force profile and then optimized through RL (Kalakrishnan et al., 2011). Prieur et al. (2012) decomposed the whole dexterous manipulation problem into a sequence of canonical-grasp-type identified in the humans. Although the introduction of human motion helps the problem, the motion of the robot is limited to these grasp types. Conversely, an "object-centric" demonstration which only demonstrated the motion of the object was adopted due to the special end-effector used in the work of Gupta et al. (2016). Also, the demonstrations can be used to per-train an initial policy (Rajeswaran et al., 2018; Alakuijala et al., 2021). For further improving the sample efficiency, Alakuijala et al. (2021) adopted residual reinforcement learning.

TABLE 5 | Overview of the dexterous manipulation solved by RL from scratch.

References	Method	Manipulator	Sensors	Environment	Tasks
Popov et al. (2017)	Improved DDPG	Jaco arm	-	Simulation only	Lego assembly
Fakoor et al. (2020)	DDPG++	ADROIT hand	-	Simulation only	Door opening
He and Ciocarlie (2021)	DisoSyn (based on PPO)	Shadow hand	-	Simulation only	Multi-tasks
Huang et al. (2021)	DDPG+HER+Multi-task learning	Shadow hand	-	Simulation only	In-hand rotation
Katyal et al. (2016)	DQN	Modular Prosthetic Limb (MPL)	-	Simulation only	In-hand manipulation
Li S. et al. (2019)	DDPG+HER	Shadow hand	-	Simulation only	Solving a 2*2*2 Rubik's Cube
Omer et al. (2021)	MPC-SAC	Dclaw and Shadow hand	-	Simulation only	Valve-turning and manipulating a cube
He et al., 2020	Soft HER	Shadow hand	-	Simulation only	Hand manipulate block and others
Xu et al. (2021)	SAC	Allegro hand	tactile sensors	Simulation only	Playing piano
Kumar et al. (2016)	RL with linear-Gaussian controllers (model-based RL)	Adroit platform	pressure sensors and piston length sensors	Simulation and real robot	Hand positioning and object manipulation
van Hoof et al. (2015)	NPREPS (van Hoof et al., 2015)	An under-actuated compliant robot hand	Tactile sensor	Real world	Rolling an object between fingertips
Nagabandi et al. (2020)	PDDM (model-based RL)	Shadow hand	Camera tracker	Real world	Baoding balls
Haarnoja et al. (2019)	SAC	Dclaw	Visual sensor	Real world	Valve rotation
Zhu H. et al. (2018)	TNPG	Dclaw and Allegro Hand	-	Real world	Valve Rotation and Door opening
Gupta et al. (2021)	MTRF	D'Hand	-	Real world	Pipe insertion and In-hand manipulation

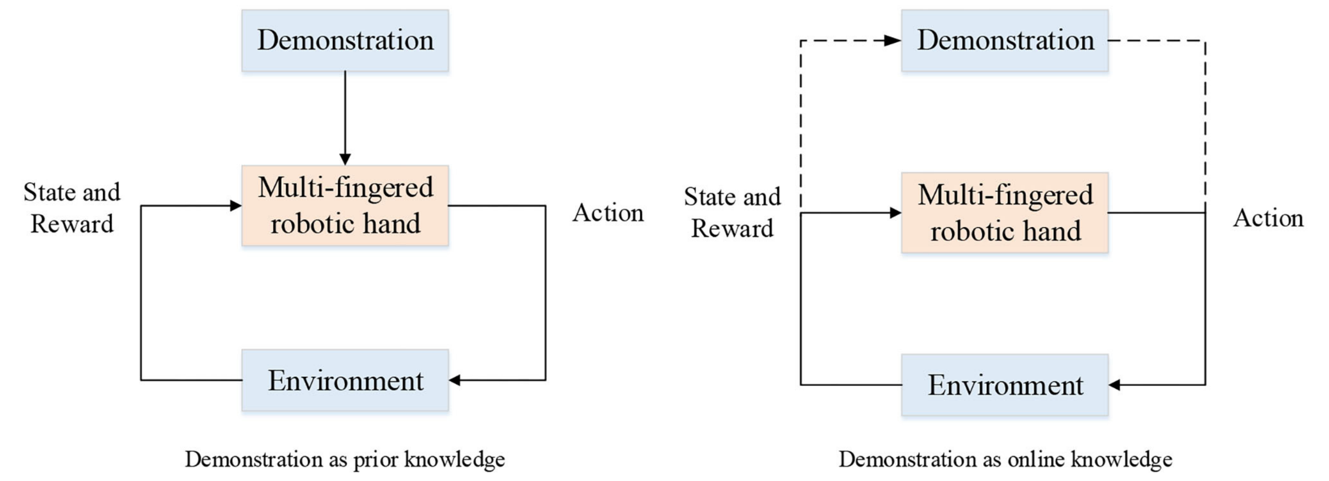

Demonstration
State and Reward
Multi-fingered robotic hand
Action
Environment
Demonstration as prior knowledge
Demonstration
State and Reward
Multi-fingered robotic hand
Action
Environment
Demonstration as online knowledge

FIGURE 6 | Two types of combination of RL and demonstration.

The demonstration data also can be stored to provide an auxiliary part in the reward function. Considering the state-action pairs trajectories are not available all the time, Radosavovic et al. (2020) proposed State-Only Imitation Learning (SOIL) where an inverse model is also learned to infer the action for the demonstrated state. An important work combining reinforcement learning and imitation learning is generative adversarial imitation learning (GAIL), which is used widely in the domain of dexterous manipulation (Zhu Y. et al., 2018) DexMV (Qin et al., 2021). Orbik et al.

(2021) adopted the inverse reinforcement learning method and improved the original algorithm to the problem that the learned rewards are strongly inclined to the demonstrated actions using statistical tools for random sample generation and reward normalization.

In the second class, the demonstrations are usually stored in the replay buffer and act as online knowledge to provide guidance. Jeong et al. (2021) used a set of waypoints (pose) tracking controllers as a suboptimal expert. The demonstration data were used in the exploration process occasionally by intertwining with the online interaction data. And, the combination of the exploration strategy and the Relative Entropy Q-Learning (REQ) algorithm called REQfSE outperformed the DDPG from demonstrations (DDPGfD) (Vecerik et al., 2018) and MPOfD (Jeong et al., 2019) on several tasks, such as single-arm stacking in the simulation environment. Garcia-Hernando et al. (2020) used the imperfect estimated hand pose as a demonstration. The action was combined between the hand pose estimation from inverse kinematics (IK) and the output of the residual policy network for imitating the hand pose in the real world more accurately. Because of its sheer volume and availability, a raw video is an appealing form of demonstration data. DexMV (Qin et al., 2021) just adopted this idea. They estimated the hand-object pose from raw video and used the estimation as demonstration data to learn robust policy with imitation learning. This work is a great beginning for further research in dexterous manipulation or any other vision-based research related to imitation learning.

According to the analysis previously, a summary of the works in this section is listed in **Table 6.**

Dexterous Manipulation From Simulation to Real Robotics

Benefiting from the parallel and powerful computations, collecting data in simulators is easier and safer than that in the real world. Therefore, learning in simulation and then transferring the learned policy to a real robot is appealing to researchers. However, the discrepancies between simulation and real robot make the transformation challenging, which are generally called "reality gaps" including dynamics differences of engines, and so on. Transforming the policy directly to the real world may cause various consequences, the lesser of which is a decline in success and the more serious of which is the instability of the system that may destroy the robotic hands or the environment. Hence, closing the reality gap is the main issue when mentioning the sim-to-real problem. For narrowing the gap, some researchers focused on building higher fidelity simulators such as MuJoCo (Todorov et al., 2012), PyBullet (Coumans and Bai, 2016), and Gazebo (Koenig and Howard, 2004). However, it is generally accepted that the improvement of simulators will not bridge the gap completely. The typical approaches for bridging the reality gap in the domain of dexterous manipulation with multi-fingered robotic hands with RL are depicted in **Figure 7** and the application of these approaches in this domain will be introduced in the following part.

The sim-to-real problem is not unique to the field of reinforcement learning or dexterous manipulation, but general problem in machine learning. The main approaches widely used for closing the reality gap are system identity, domain randomization, and transfer learning including domain adaptation and progressive networks. However, on account of that the models of multi-fingered robotic hands and the complex environment are impossible to be accurately built in the simulators. The simplest system identity method is not desirable and other approaches must be considered. Instead of building an accurate model of the real world in system identity, the main idea of domain randomization is to randomize the simulation with disturbance. The elements can be randomized and include many aspects which can be roughly divided into parts visual randomization and dynamic randomization. For instance, the randomization of lighting, textures of the object, and the positions of the cameras belong to visual randomization, and the randomization of surface friction coefficients, the contact model, and the object mass belong to dynamics randomization. Through exposure to various environments, the learner trained in simulation can adapt to a wide range of environments. So for the learner, the real world is just a disturbed environment. More details of the sim-to-real problem can be seen in Zhu et al. (2021).

The idea of randomization is widely adopted in the sim-to-real problem of dexterous manipulation (Allshire et al., 2021). For instance, in the work of Zhu H. et al. (2018), only visual randomizations were adopted for zero-shot transfer from simulation to reality. Unlike learning policies robust to senses with high variation mentioned before, Kumar et al. (2019) focused on the variation of object appearance and geometry such as object mass, friction coefficients between the fingers and object, PD gains of the robot, and damping coefficients of the robot joints. Visual sensing is used to abstract away the uncertainties into a succinct set of geometric features and tactile sensors are adopted to compensate for the inaccurate approximation. After training in the simulation, a zero-shot transfer is achieved on the real robot for a grasping task. Similarly, the idea of randomization of friction, object mass, and object scale was also adopted by Allshire et al. (2021), where the training process was carried out in IsaacGym (Liang et al., 2018). The notable work accomplished by OpenAI (Andrychowicz et al., 2020) also adopted the approach of domain randomization to transfer the policy learned in the MuJoCo simulator to a real Shadow hand. Apart from visual randomizations and physics randomizations, a lot of other randomizations were adopted. Through extensive randomizations, the learned policy got a great performance in the real robot system without any fine-tuning. The success of this work demonstrates that the gap between the simulation and reality can be narrowed to a usable level. Later, they improved the algorithm to solve a more complicated task of solving a Rubik's cube (Open et al., 2019). The concept of domain randomization was also considered, however, they improved it for a better format, namely automatic domain randomization (ADR). The main improvement compared to classic domain randomization lies in the automatic change of the distribution ranges leaving out tedious manual tunning. Furthermore, unlike

TABLE 6 | Overview of the dexterous manipulation solved by RL with a demonstration.

References	Method	Manipulator	Sensors	Environment	Form of demonstration	Tasks
Qin et al. (2021)	DexMV	Adroit Hand	-	Simulation only	Raw video	Relocating, pouring and placing inside
Zhu H. et al., 2018	DAPG	Dclaw and Allegro Hand	-	Real robot	kinesthetic teaching	Valve Rotation, Valve Rotation and Door opening
Orbik et al. (2021)	IRL	Adroit Hand	-	Simulation only	CyberGlove	Object relocation, tool use, in-hand manipulation and door opening
Rajeswaran et al. (2018)	DAPG	Adroit hand	-	Simulation only	CyberGlove	Object relocation, tool use, in-hand manipulation and door opening
Gupta et al. (2016)	Learning from demonstrations algorithm based on the GPS	RBO Hand 2	Phase space Impulse system	Real robot	LED marker tracking the motion of the object demonstrated by human	Turning a valve, pushing beads on an abacus, and grasping a bottle from a table
Jeong et al. (2021)	REQfSE	Bimanual Shadow Hand	-	Simulation only	Waypoint controllers	LEGO stacking
Alakuijala et al. (2021)	RRLfD	Adroit Hand	-	Simulation only	Script or a previously trained RL agent	Object relocation, tool use, in-hand manipulation and door opening
Radosavovic et al. (2020)	SOIL	Adroit Hand	-	Simulation only	virtual reality headset and a motion capture glove	Object relocation, tool use, in-hand manipulation and door opening

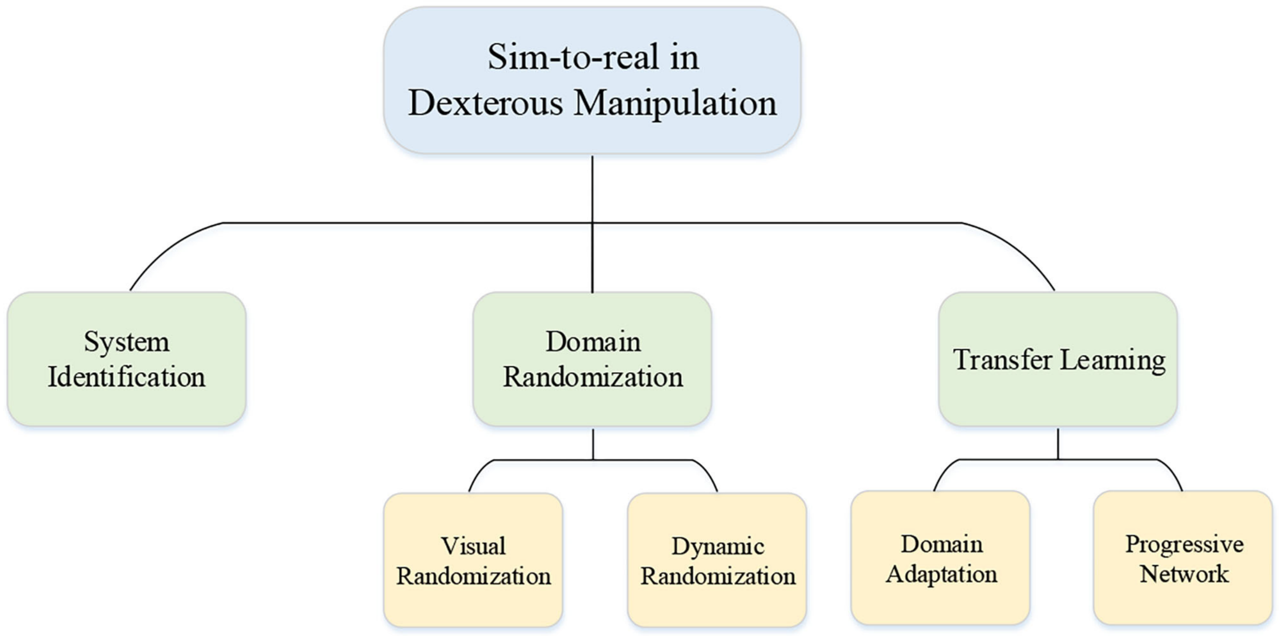


FIGURE 7 | Category of approaches for sim-to-real in this domain.

TABLE 7 | Overview of the dexterous manipulation from simulation to reality.

References	RL	Sim2Real	Manipulator	Simulator	Sensor	Task
Andrychowicz et al. (2020)	PPO	Domain randomization	Shadow hand	MuJoCo physics engine (Todorov et al., 2012)	3D tracking system and RGB cameras	Manipulating a block
Open et al. (2019)	PPO	Automatic domain randomization	Shadow hand	MuJoCo physics engine (Todorov et al., 2012)	3D tracking system and RGB cameras	Solving a Rubik's cube
Zhu Y. et al. (2018)	A method based on GAIL(IL)+PPO(RL)	Domain randomization	Jaco arm	MuJoCo physics engine (Todorov et al., 2012)	RGB cameras	Block lifting and stacking
Kumar et al. (2019)	Contextual RL (PPO)	Domain randomization	Allegro hand	-	RGB cameras and Tactile sensor	Grasping
Allshire et al. (2021)	PPO	Domain randomization	TriFinger	IsaacGym	RGB cameras	In-hand manipulation
Rusu et al. (2017)	A3C	Progressive net	Jaco arm	MuJoCo physics engine (Todorov et al., 2012)	RGB cameras	Reaching to a visual target
Fernandes Veiga et al. (2020)	Hierarchical control (RL+ tactile feedback control)	Hierarchical RL	Allegro hands	PyBullet Coumans and Bai (2016) simulation environment	Tactile sensor	In-hand manipulation

fixed distribution ranges in classic domain randomization, the distribution ranges are allowed to change in ADR instead.

The intuition of transfer learning is leveraging the data from a source domain where the data are abundant and sufficient to help learn a robust policy in the target domain with little data. The progressive network proposed by Deepmind (Rusu et al., 2016) is a unique structure of a neural network with the ability to use the knowledge of the previous task for the new task without catastrophic forgetting. Later, they adopted this idea for robot manipulation (Rusu et al., 2017). Also, some researchers focused on the RL algorithm itself such as hierarchical decomposition RL (Fernandes Veiga et al., 2020) to bridge the reality gap. Considering that the privileged state information is not available in reality, researchers usually used rendered pictures as observation (Open et al., 2019; Andrychowicz et al., 2020). However, the accurate privileged state information in a simulator can accelerate the training process and get a better policy. An idea of teacher-student training which transfers the better teacher policy to a student policy that only uses sensory inputs was adopted in (Chen et al., 2021) for accelerating the training process in real world. A summary of the works in this section is listed in **Table 7**.

CHALLENGES AND FUTURE RESEARCH DIRECTIONS

Although the methods mentioned in this paper already solved part of the dexterous manipulation problems, we are still a long way from making the robotic hands as dexterous as human hands. And the complexity of the multi-fingered robotic hand system, such as uncertain models, dimensional disaster has restricted the development of RL in dexterous manipulation with multi-fingered robotic hand domain. In general, the challenges the community is facing in this domain are as follows:

- *Sample inefficiency:* The demand of more data limits the tasks which can be solved by RL from scratch to a narrow scope.
- *Tradeoff between exploration and exploitation:* Through exploration, the robot can get more information about the environment, but random behavior may not get rewards in tasks with sparse rewards, which would not make the algorithm converge. On the other hand, exploitation gives more knowledge about the environment to make the best decision, but the deficiency of information may lead to a locally optimal solution. Therefore, two questions should be answered: how to explore efficiently and effectively and when to transition from exploration to exploitation.
- *Choosing of suitable manipulator:* High stiffness improves precision but lacks flexibility and may damage the environment, whereas low stiffness (i.e., soft robotic fingers) improves robustness but suffers from inaccuracy. Furthermore, there is a tradeoff between dexterity and control simplicity.
- *Reality gap:* Despite the methods such as domain randomization mitigating the gap to the extent of one-shot transferring, the reality gap is also a problem that cannot be ignored.
- *High cost of time and resources:* A long time is required for obtaining a robust policy in terms of large-scale experiments. Furthermore, the multi-fingered robotic hands are so expensive and fragile that maintaining and repairing these robotic hands costs much. The immense requirement keeps such success at the laboratory level.
- *Poor generalization ability:* In general, the learned policy only fits to the specific task and robot, generalizing the policy to different robotic hands and tasks remains challenging.

- *Hardware limitations:* The high demands on the sensors makes it challenging to achieve the dexterity of a human hand. Moreover, the rigid plastic and metal components of the current robotic hands are the main reasons for the lack of dexterity. Although a variety of commercial products are available, their touch sensors are rigid and their placement is limited to the fingertips and along the limb segments, which is not desirable.
- *Complex manipulation is still unavailable:* Although some simple tasks such as pick-and-place, throwing, sliding, pivoting, and pushing can be done, some more complex tasks, especially those that change the shape of objects (cutting, crushing) are unavailable. A model of the deformation and advanced perception to monitor the changes is required.

To meet the challenges and accelerate the process of robotic hands intelligence, the future directions for researchers can be summarized below:

- ⧫ *More advanced simulators:* Although some great simulators can be as fast as realistic or even faster in many cases, the existing simulators have certain limitations for emulating some elements of the environment. The more advanced the simulators are, the better performance in transferring the policy learned in simulation to reality. Furthermore, more manipulation scenarios are more desirable.
- ⧫ *Fusion of sensors:* For more accurate information about the system, the visual sensing information used widely in the previous works is not sufficient, so multimodal sensory signals which include, but are not limited to, tactile and temperature signals should be used to represent the state of the system.
- ⧫ *Improvement of the algorithm:* The rewards in the existing algorithms are typically designed carefully and only simple tasks such as reaching and pushing can be accomplished with sparse rewards. For this problem, informed exploration may be helpful. Furthermore, the adaptation to the variations of both the robot variations and variations in the environment is essential for working gracefully. Therefore, more sophisticated methodologies must be found for dealing with these problems and accelerating the training process.
- ⧫ *Semantic understanding:* Learning to understand the environment and the task and following the human order are also vital skills for a robot to work with more intelligence. For a given order from voice or other forms, a robot should know what to do and how to do the task.
- ⧫ *Improvement of robotic hands:* Although there have been many robotic hands in this domain, the limited dexterity of the simple end-effector and the fragility and characteristics that are not conducive to controlling the complex dexterous multi-fingered hands hinder the development of the domain. The tradeoff of the dexterity and the complexity of control should be balanced.
- ⧫ *Manipulation in media such as water or oil:* The existing successful examples of dexterous manipulation are all in the air. However, for some special tasks, such as underwater operation, the ability to manipulate in the water is especially important.
- ⧫ *Deeper study in basic theoretical:* Currently, the model of soft point-contact and stability rules for both point contacts and surface contacts, which are vital for modeling the system, are not available. Although the model is not essential for a learning-based approach, the emphasis on theory may be conducive for a better simulator.

CONCLUSION

In this paper, we present a brief overview of the reinforcement learning solutions for dexterous manipulation, focusing mainly on reinforcement learning, reinforcement learning from demonstration, and transfer learning from simulation to reality. The application of reinforcement learning in dexterous manipulation with the multi-fingered robotic hand is mostly hampered by the high cost of collecting sufficient data for a great policy. At present, the common and effective ways for mitigating data inefficiency issues are learning from demonstration and transferring the learned policy in simulation to the real world. However, compared with the tasks that humans can handle easily, what the multi-fingered robotic hands can do is still very limited. Despite this, we believe that the reinforcement learning-based solution can do a lot as the research goes further.

AUTHOR CONTRIBUTIONS

PW contributed to the conception of the study and revised the manuscript critically for important intellectual content. CY collected related literature and wrote the manuscript. Both authors reviewed the results and approved the final version of the manuscript.

FUNDING

This work was supported in part by the National Natural Science Foundation of China under Grants (91748131, 62006229, and 61771471), in part by the Strategic Priority Research Program of Chinese Academy of Science under Grant (XDB32050106), and in part by the InnoHK Project.

REFERENCES

Alakuijala, M., Dulac-Arnold, G., Mairal, J., Ponce, J., and Schmid, C. (2021). Residual reinforcement learning from demonstrations. ArXiv210608050 Cs.

Al-Gallaf, E., and Warwick, K. (1995). Force distribution in manipulation by a robot hand with equality and inequality constraints. *Mechatronics.* 5, 561–583. doi: 10.1016/0957-4158(95)00017-Y

Allegro hand. "Allegro hand," Available online at: https://github.com/simlabrobotics/allegro_hand_ros

Allshire, A., Mittal, M., Lodaya, V., Makoviychuk, V., Makoviichuk, D., Widmaier, F., et al. (2021). Transferring dexterous manipulation from GPU simulation to a remote real-world trifinger. ArXiv210809779 Cs.

Anderson, R. J., and Spong, M. W. (1988). Hybrid impedance control of robotic manipulators. *IEEE J. Robot. Autom*. 4, 549–556. doi: 10.1109/56.20440

Andrychowicz, O. M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., et al. (2020). Learning dexterous in-hand manipulation. *Int. J. Robot. Res*. 39, 3–20. doi: 10.1177/0278364919887447

Bicchi, A. (1995). On the closure properties of robotic grasping. *Int. J. Robot. Res*. 14, 319–334. doi: 10.1177/027836499501400402

Bicchi, A. (2000). Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity. *IEEE Trans. Robot. Autom*. 16, 652–662. doi: 10.1109/70.897777

Billard, A. (2019). Trends and challenges in robot manipulation. *Science.* 364. doi: 10.1126/science.aat8414

Bruyninckx, H., Demey, S., and Kumar, V. (1998). Generalized stability of compliant grasps. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation* (Cat. No.98CH36146). vol. 3. p. 2396–2402.

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. (2019). Sample-efficient reinforcement learning with stochastic ensemble value expansion. ArXiv180701675 Cs Stat.

Buss, M., Hashimoto, H., and Moore, J. B. (1996). Dextrous hand grasping force optimization. *IEEE Trans. Robot. Autom*. 12, 406–418. doi: 10.1109/70.499823

Charlesworth, H. J., and Montana, G. (2021). Solving Challenging Dexterous Manipulation Tasks With Trajectory Optimisation and Reinforcement Learning. In: *Proceedings of the 38th International Conference on Machine Learning*. p. 1496–1506.

Chen, C., Culbertson, P., Lepert, M., Schwager, M., and Bohg, J. (2021a). Trajectotree: trajectory optimization meets tree search for planning multi-contact dexterous manipulation. ArXiv210914088 Cs. doi: 10.1109/IROS51168.2021.9636346

Chen, C., Srinivasan, K., Zhang, J., Zhang, J., Shao, L., Yuan, S., et al. (2021b). Dexterous manipulation primitives for the real robot challenge. ArXiv210111597 Cs.

Chen, T., Xu, J., and Agrawal, P. (2021). A system for general in-hand object re-orientation. ArXiv211103043 Cs.

Coumans, E., and Bai, Y. (2016). *Pybullet, a Python Module for Physics Simulation for Games*. Robotics and machine learning.

Cui, J., and Trinkle, J. (2021). Toward next-generation learned robot manipulation. *Sci. Robot.* 6, eabd9461. doi: 10.1126/scirobotics.abd9461

Cutkosky, M. R. (1989). On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Trans. Robot. Autom*. 5, 269–279. doi: 10.1109/70.34763

Deisenroth, M., and Rasmussen, C. (2011). PILCO: a model-based and data-efficient approach to policy search. In*: Proceedings of the 28th International Conference on Machine Learning*. p. 465–472.

Duan, H., Wang, P., Huang, Y., Xu, G., Wei, W., and Shen, X. (2021). Robotics dexterous grasping: The methods based on point cloud and deep learning. *Front. Neurorobot*. 15, 658280. doi: 10.3389/fnbot.2021.658280

Eberman, B., and Salisbury, J. K. (1994). Application of change detection to dynamic contact sensing. *Int. J. Robot. Res*. 13, 369–394. doi: 10.1177/027836499401300501

Fakoor, R., Chaudhari, P., and Smola, A. J. (2020). DDPG++: striving for simplicity in continuous-control off-policy reinforcement learning. ArXiv200615199 Cs Stat.

Falco, P., Attawia, A., Saveriano, M., and Lee, D. (2018). On policy learning robust to irreversible events: an application to robotic in-hand manipulation. *IEEE Robot. Autom. Lett*. 3, 1482–1489. doi: 10.1109/LRA.2018.2800110

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. (2018). Model-based value estimation for efficient model-free reinforcement learning. ArXiv180300101 Cs Stat.

Fernandes Veiga, F., Akrour, R., and Peters, J. (2020). Hierarchical tactile-based control decomposition of dexterous in-hand manipulation tasks. *Front. Robot. AI*. 7, 521448. doi: 10.3389/frobt.2020.521448

Ferrari, C., and Canny, J. F. (1992). Planning optimal grasps. In: *ICRA*. p. 6.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In: *Proceedings of the 35th International Conference on Machine Learning.* pp. 1587–1596.

Garcia-Hernando, G., Johns, E., and Kim, T.-., K. (2020). Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning. ArXiv200803285 Cs. doi: 10.1109/IROS45743.2020.9340947

Goldenberg, A. A. (1988). Implementation of force and impedance control in robot manipulators. In: *1988 IEEE International Conference on Robotics and Automation Proceedings*. vol. 3. p. 1626–1632.

Gupta, A., Eppner, C., Levine, S., and Abbeel, P. (2016). Learning dexterous manipulation for a soft robotic hand from human demonstrations. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* p. 3786–3793. doi: 10.1109/IROS.2016.7759557

Gupta, A., Yu, J., Zhao, T. Z., Kumar, V., Rovinsky, A., Xu, K., et al. (2021). Reset-free reinforcement learning via multi-task learning: learning dexterous manipulation behaviors without human intervention. ArXiv210411203 Cs. doi: 10.1109/ICRA48506.2021.9561384

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

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., et al. (2019). Soft actor-critic algorithms and applications. ArXiv181205905 Cs Stat.

Handa, A., Van Wyk, K., Yang, W., Liang, J., Chao, Y. W., et al. (2019). DexPilot: vision based teleoperation of dexterous robotic hand-arm system. ArXiv191003135 Cs. doi: 10.1109/ICRA40945.2020.9197124

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. (2018). Learning an embedding space for transferable robot skills. In: *Proceedings the International Conference on Learning Representations*.

He, Q., Zhuang, L., and Li, H. (2020). Soft hindsight experience replay. ArXiv200202089 Cs.

He, Z., and Ciocarlie, M. (2021). Discovering synergies for robot manipulation with multi-task reinforcement learning. ArXiv211001530 Cs.

Hogan, N. (1984). Impedance control: an approach to manipulation. In: 1984 *American Control Conference.* IEEE, San Diego, CA, USA. p. 304–313. doi: 10.23919/ACC.1984.4788393

Hsu, P., Li, Z., and Sastry, S. (1988). On grasping and coordinated manipulation by a multifingered robot hand. In: *Proceedings. 1988 IEEE International Conference on Robotics and Automation.* Philadelphia, PA, USA, pp. 384–389. doi: 10.1109/ROBOT.1988.12078

Hua, J., Zeng, L., Li, G., and Ju, Z. (2021). Learning for a robot: deep reinforcement learning, imitation learning, transfer learning. *Sensors.* 21, 1278. doi: 10.3390/s21041278

Huang, W., Mordatch, I., Abbeel, P., and Pathak, D. (2021). Generalization in dexterous manipulation via geometry-aware multi-task learning. ArXiv211103062 Cs Eess.

Hyde, J. M., and Cutkosky, M. R. (1998). A phase management framework for event-driven dextrous manipulation. *IEEE Trans. Robot. Autom*. 14, 978–985. doi: 10.1109/70.736781

Hyde, J. M., Tremblay, M. R., and Cutkosky, M. R. (1997). An object-oriented framework for event-driven dextrous manipulation. In: Khatib, O., Salisbury, J.K. (Eds.), *Experimental Robotics IV, Lecture Notes in Control and Information Sciences.* Springer, Berlin, Heidelberg, p. 51–61. doi: 10.1007/BFb0035196

Jacobsen, S., Iversen, E., Knutti, D., Johnson, R., and Biggers, K. (1986). Design of the Utah/M.I.T. Dextrous Hand. In: *1986 IEEE International Conference on Robotics and Automation Proceedings.* p. 1520–1532.

Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: model-based policy optimization. ArXiv190608253 Cs Stat.

Jeong, R., Aytar, Y., Khosid, D., Zhou, Y., Kay, J., Lampe, T., et al. (2019). Self-supervised sim-to-real adaptation for visual robotic manipulation. ArXiv191009470 Cs. doi: 10.1109/ICRA40945.2020.9197326

Jeong, R., Springenberg, J. T., Kay, J., Zheng, D., Zhou, Y., Galashov, A., et al. (2021). Learning dexterous manipulation from suboptimal experts. ArXiv201008587 Cs.

Johansson, R., and Westling, G. (1991). Afferent signals during manipulative tasks in humans. p. 25–48. doi: 10.1007/978-1-349-11597-6_3

Kalakrishnan, M., Righetti, L., Pastor, P., and Schaal, S. (2011). Learning force control policies for compliant manipulation. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, San Francisco, CA. p. 4639–4644. doi: 10.1109/IROS.2011.6095096

Kao, I., and Cutkosky, M. R. (1992). Quasistatic manipulation with compliance and sliding. *Int. J. Robot. Res.* 11, 20–40. doi: 10.1177/027836499201100102

Katyal, K. D., Staley, E. W., Johannes, M. S., Wang, I. J., Reiter, A., et al. (2016). In-hand robotic manipulation via deep reinforcement learning. In: *Proceedings of the Workshop on Deep Learning for Action and Interaction, in Conjunction with Annual Conference on Neural Information Processing Systems*. Barcelona, Spain.

Kelly, R., and Carelli, R. (1988). Unified approach to adaptive control of robotic manipulators. In: *Proceedings of the 27th IEEE Conference on Decision and Control.* vol. 2. p. 1598–1603.

Kelly, R., Carelli, R., Amestegui, M., and Ortega, R. (1989). On adaptive impedance control of robot manipulators. In: *1989 International Conference on Robotics and Automation Proceedings.* vol. 1. p. 572–577.

Kirkpatrick, D., Mishra, B., and Yap, C. K. (1992). Quantitative Steinitz's theorems with applications to multifingered grasping. *Discr. Comput. Geom.*. 7, 295–318. doi: 10.1007/BF02187843

Koenig, N., and Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566).* vol. 3. p. 2149–2154.

Kumar, V., Hermans, T., and Fox, D., Birchfield, S., Tremblay, J. (2019). Contextual reinforcement learning of visuo-tactile multi-fingered grasping policies. arXiv preprint arXiv:1911.09233.

Kumar, V., Tassa, Y., Erez, T., and Todorov, E. (2014). Real-time behaviour synthesis for dynamic hand-manipulation. In: *2014 IEEE International Conference on Robotics and Automation (ICRA).* p. 6808–6815. doi: 10.1109/ICRA.2014.6907864

Kumar, V., Todorov, E., and Levine, S. (2016). Optimal control with learned local models: application to dexterous manipulation. In: *2016 IEEE International Conference on Robotics and Automation (ICRA).* p. 378–383. doi: 10.1109/ICRA.2016.7487156

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature.* 521, 436–444. doi: 10.1038/nature14539

Levine, S., and Abbeel, P. (2014). Learning neural network policies with guided policy search under unknown dynamics. In: *In Advances in Neural Information Processing Systems.* p. 1071–1079.

Li, M., Yin, H., Tahara, K., and Billard, A. (2014). Learning object-level impedance control for robust grasping and dexterous manipulation. In: *2014 IEEE International Conference on Robotics and Automation (ICRA).* pp. 6784–6791. doi: 10.1109/ICRA.2014.6907861

Li, S., Jiang, J., Ruppel, P., Liang, H., Ma, X., Hendrich, N., et al. (2020). A mobile robot hand-arm teleoperation system by vision and IMU. ArXiv200305212 Cs. doi: 10.1109/IROS45743.2020.9340738

Li, S., Ma, X., Liang, H., Görner, M., Ruppel, P., Fang, B., et al. (2019). Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network. ArXiv180906268 Cs. doi: 10.1109/ICRA.2019.8794277

Li, T., Xi, W., Fang, M., Xu, J., and Meng, M. Q. H. (2019). Learning to solve a rubik's cube with a dexterous hand. ArXiv190711388 Cs. doi: 10.1109/ROBIO49542.2019.8961560

Li, Y., Wei, W., Li, D., and Wang, P. (2022). "HGC-Net: Deep anthropomorphic hand grasping in clutter," in *IEEE International Conference on Robotics and Automation*.

Liang, J., Makoviychuk, V., Handa, A., Chentanez, N., Macklin, M., and Fox, D. (2018). GPU-accelerated robotic simulation for distributed reinforcement learning. In: *Proceedings of The 2nd Conference on Robot Learning*. pp. 270–282.

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

Lin, Q., and Burdick, J. W. (1999). A task-dependent approach to minimum-deflection fixtures. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C).* vol. 2. pp. 1562–1569.

Lin, Q., Burdick, J. W., and Rimon, E. (2000). A stiffness-based quality measure for compliant grasps and fixtures. *IEEE Trans. Robot. Autom.* 16, 675–688. doi: 10.1109/70.897779

Liu, H., Wu, K., Meusel, P., Seitz, N., Hirzinger, G., Jin, M. H., et al. (2008). Multisensory five-finger dexterous hand: the DLR/HIT Hand II. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems.* pp. 3692–3697. doi: 10.1109/IROS.2008.4650624

Liu, Z., Luo, X., and Wang, Z. (2021). Convergence analysis of single latent factor-dependent, nonnegative, and multiplicative update-based nonnegative latent factor models. *IEEE Trans. Neural Netw. Learn. Syst.* 32, 1737–1749. doi: 10.1109/TNNLS.2020.2990990

Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E., and Mordatch, I. (2019). Plan online, learn offline: efficient learning and exploration via model-based control. ArXiv181101848 Cs Stat.

Luo, X., Yuan, Y., Chen, S., Zeng, N., and Wang, Z. (2020). Position-transitional particle swarm optimization-incorporated latent factor analysis. In: *IEEE Transactions on Knowledge and Data Engineering.* p. 1–1. doi: 10.1109/TKDE.2020.3033324

Ma, R. R., and Dollar, A. M. (2011). On dexterity and dexterous manipulation. In: *2011 15th International Conference on Advanced Robotics (ICAR).* pp. 1–7. doi: 10.1109/ICAR.2011.6088576

Melnik, A., Lach, L., Plappert, M., Korthals, T., Haschke, R., and Ritter, H. (2019). Tactile sensing and deep reinforcement learning for in-hand manipulation *tasks*. In: *IROS Workshop on Autonomous Object Manipulation*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., et al. (2016). Asynchronous methods for deep reinforcement learning. In: *International Conference on Machine Learning.*

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

Montana, D. J. (1988). The kinematics of contact and grasp. *Int. J. Robot. Res.* 7, 17–32. doi: 10.1177/027836498800700302

Mordatch, I., Popovic, Z., and Todorov, E. (2012a). Contact-invariant optimization for hand manipulation. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* p. 137–144.

Mordatch, I., Todorov, E., and Popovi,ć, Z. (2012b). Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.* 31, 1–8. doi: 10.1145/2185520.2185539

Mori, S., Tangkaratt, V., Zhao, T., Morimoto, J., and Sugiyama, M. (2013). Model-based policy gradients with parameter-based exploration by least-squares conditional density estimation. ArXiv13075118 Cs Stat. doi: 10.1016/j.neunet.2014.06.006

Nagabandi, A., Konolige, K., Levine, S., and Kumar, V. (2020). Deep dynamics models for learning dexterous manipulation. In: *Proceedings of the Conference on Robot Learning*. p. 1101–1112.

Nahon, M. A., and Angeles, J. (1991). Optimization of dynamic forces in mechanical hands. *J. Mech. Des.* 113, 167–173. doi: 10.1115/1.2912765

Nakamura, Y., Nagai, K., and Yoshikawa, T. (1989). Dynamics and stability in coordination of multiple robotic mechanisms. *Int. J. Robot. Res.* 8, 44–61. doi: 10.1177/027836498900800204

Ng, A. Y., and Jordan, M. I. (2013). *PEGASUS: A Policy Search Method for Large MDPs and POMDPs.*

Oh, J., Singh, S., and Lee, H. (2017). Value prediction network. In: *Advances in Neural Information Processing Systems.* Curran Associates, Inc.

Okamura, A. M., Smaby, N., and Cutkosky, M. R. (2000). An overview of dexterous manipulation. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065). pp. 255–262 vol. 1.

Omer, M., Ahmed, R., Rosman, B., and Babikir, S. F. (2021). Model Predictive-Actor Critic Reinforcement Learning for Dexterous Manipulation. In: *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE).* pp. 1–6. doi: 10.1109/ICCCEEE49695.2021.9429677

Open AI., Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., et al. (2019). Solving rubik's cube with a robot hand. ArXiv191007113 Cs Stat.

Orbik, J., Agostini, A., and Lee, D. (2021). Inverse reinforcement learning for dexterous hand manipulation. In: *2021 IEEE International Conference on Development and Learning (ICDL).* p. 1–7. doi: 10.1109/ICDL49984.2021.9515637

Popov, I., Heess, N., Lillicrap, T., Hafner, R., Barth-Maron, G., Vecerik, M., et al. (2017). Data-efficient deep reinforcement learning for dexterous manipulation. arXiv preprint arXiv:1704.03073.

Prattichizzo, D., Pozzi, M., and Malvezzi, M. (2020). Dexterous manipulation. In: Ang, M.H., Khatib, O., Siciliano, B. (Eds.), *Encyclopedia of Robotics.* Springer Berlin Heidelberg, Berlin, Heidelberg, p. 1–8. doi: 10.1007/978-3-642-41610-1_180-1

Prieur, U., Perdereau, V., and Bernardino, A. (2012). Modeling and planning high-level in-hand manipulation actions from human knowledge and active learning from demonstration. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 1330–1336. doi: 10.1109/IROS.2012.6386090

Qin, Y., Wu, Y.-., H., Liu, S., Jiang, H., Yang, R., et al. (2021). DexMV: imitation learning for dexterous manipulation from human videos. ArXiv210805877 Cs.

Radosavovic, I., Wang, X., Pinto, L., and Malik, J. (2020). State-only imitation learning for dexterous manipulation. ArXiv200404650 Cs Stat. doi: 10.1109/IROS51168.2021.9636557

Raibert, M. H., and Craig, J. J. (1981). Hybrid position/force control of manipulators. *J. Dyn. Syst. Meas. Control.* 103, 126–133. doi: 10.1115/1.3139652

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., et al. (2018). Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. ArXiv170910087 Cs. doi: 10.15607/RSS.2018.XIV.049

Ramírez, J., Yu, W., and Perrusquía, A. (2021). Model-free reinforcement learning from expert demonstrations: a survey. *Artif. Intell. Rev.* 55, 3213–3241. doi: 10.1007/s10462-021-10085-1

Roa, M. A., and Suárez, R. (2009). Finding locally optimum force-closure grasps. Robot. *Comput.-Integr. Manuf.* 25, 536–544. doi: 10.1016/j.rcim.2008.02.008

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., et al. (2016). Progressive neural networks. ArXiv160604671 Cs.

Rusu, A. A., Večerík, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. (2017). Sim-to-real robot learning from pixels with progressive nets. In: *Proceedings of the 1st Annual Conference on Robot Learning*. p. 262–270.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. ArXiv151105952 Cs.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In: *International Conference on Machine Learning*. p. 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. ArXiv170706347 Cs.

Shadow Hand (2005). "Shadow dexterous Hand, (2005). Available online at: https://www.shadowrobot.com/dexterous-hand-series/

Shang, M., Luo, X., Liu, Z., Chen, J., Yuan, Y., and Zhou, M. (2019). Randomized latent factor model for high-dimensional and sparse matrices from industrial applications. *IEEECAA J. Autom. Sin*. 6, 131–141. doi: 10.1109/JAS.2018.7511189

Spong, M. W., Hutchinson, S., and Vidyasagar, M. (2006). *Robot Modeling and Control*. Vol. 3. New York: Wiley.

Stansfield, S. A. (1991). Robotic grasping of unknown objects: a knowledge-based approach. *Int. J. Robot. Res*. 10, 314–326. doi: 10.1177/027836499101000402

Sundaralingam, B., and Hermans, T. (2017). Relaxed-rigidity constraints: in-grasp manipulation using purely kinematic trajectory optimization. *Planning*. 6, 7. doi: 10.15607/RSS.2017.XIII.015

Sundaralingam, B., and Hermans, T. (2018). Geometric in-hand regrasp planning: alternating optimization of finger gaits and in-grasp manipulation. ArXiv180404292 Cs. doi: 10.1109/ICRA.2018.8460496

Sutton, R. S. (1990). Dyna, an integrated architecture for learning, planning, and reacting. In: *Proceedings of the SevenLh International Conference on Machine Learning*. pp. 216–224. doi: 10.1016/B978-1-55860-141-3.50030-4

Todorov, E., Erez, T., and Tassa, Y. (2012). MuJoCo: a physics engine for model-based control. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 5026–5033. doi: 10.1109/IROS.2012.6386109

van Hasselt, H., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning. ArXiv150906461 Cs.

van Hoof, H., Hermans, T., Neumann, G., and Peters, J. (2015). Learning robot in-hand manipulation with tactile features. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, Seoul, South Korea. p. 121–127. doi: 10.1109/HUMANOIDS.2015.7363524

Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., et al. (2018). Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. ArXiv170708817 Cs.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. ArXiv151106581 Cs.

Wei, W., Li, D., Wang, P., Li, Y., Li, W., Luo, Y., and Zhong, J. (2022). DVGG: Deep variational grasp generation for dextrous manipulation. *IEEE Robot. Autom. Lett.* 7, 1659–1666. doi: 10.1109/LRA.2022.3140424

Wei, W., Luo, Y., Li, F., Xu, G., Zhong, J., Li, W., and Wang, P. (2021). "Gpr: Grasp pose refinement network for cluttered scenes," in 2021 *IEEE International Conference on Robotics and Automation* (IEEE), 4295–4302.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* 8, 229–256. doi: 10.1007/BF00992696

Wu, D., Luo, X., Shang, M., He, Y., Wang, G., and Wu, X. (2020). A data-characteristic-aware latent factor model for web services QoS prediction. In: *IEEE Transactions on Knowledge and Data Engineering.* p. 1–1. doi: 10.1109/TKDE.2020.3014302

Wu, H., Luo, X., and Zhou, M. (2020). Advancing non-negative latent factorization of tensors with diversified regularizations. In: *IEEE Transactions on Services Computing*. p. 1. doi: 10.1109/TSC.2020.2988760

Wüthrich, M., Widmaier, F., Grimminger, F., Akpo, J., Joshi, S., Agrawal, V., et al. (2021). TriFinger: an open-source robot for learning dexterity. ArXiv200803596 Cs.

Xiao, D., Ghosh, B. K., and Ning Xi, T. (2000). Sensor-based hybrid position/force control of a robot manipulator in an uncalibrated environment. *IEEE Trans. Control Syst. Technol*. 8, 635–645. doi: 10.1109/87.852909

Xu, H., Luo, Y., Wang, S., Darrell, T., and Calandra, R. (2021). Towards learning to play piano with dexterous hands and touch. ArXiv210602040 Cs Stat.

Yoneda, T., Schaff, C., Maeda, T., and Walter, M. (2021). Grasp and motion planning for dexterous manipulation for the real robot challenge. ArXiv210102842 Cs.

Zahlner, S., Hirschmanner, M., Patten, T., and Vincze, M. (n.d.). Teleoperation system for teaching dexterous manipulation.

Zhang, T., and Mo, H. (2021). Reinforcement learning for robot research: a comprehensive review and open issues. *Int. J. Adv. Robot. Syst*. 18. doi: 10.1177/17298814211007305

Zhu, H., Gupta, A., Rajeswaran, A., Levine, S., and Kumar, V. (2018). Dexterous manipulation with deep reinforcement learning: efficient, general, and low-cost. ArXiv181006045 Cs. doi: 10.1109/ICRA.2019.8794102

Zhu, W., Guo, X., Owaki, D., Kutsuzawa, K., and Hayashibe, M. (2021). A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots. In: *IEEE Transactions on Neural Networks and Learning Systems*. 1–16. doi: 10.1109/TNNLS.2021.3112718

Zhu, Y., Wang, Z., Merel, J., Rusu, A., Erez, T., Cabi, S., et al. (2018). Reinforcement and imitation learning for diverse visuomotor skills. ArXiv180209564 Cs. doi: 10.15607/RSS.2018.XIV.009

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

OPINION
published: 29 April 2022
doi: 10.3389/fnbot.2022.846866

Peripheral Nervous System Interfaces: Invasive or Non-invasive?

Claudio Castellini [1,2]*

[1] *Chair of Medical Robotics, Friedrich-Alexander Universität Erlangen-Nürnberg, Erlangen, Germany,* [2] *The Adaptive Bio-Interfaces Group, German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Oberpfaffenhofen, Germany*

Keywords: neurorobotics, machine learning, human-robot interaction, intent detection, somatosensory feedback

OPEN ACCESS

Edited by:
Florian Röhrbein, Technische Universität Chemnitz, Germany

Reviewed by:
Solaiman Shokur, Swiss Federal Institute of Technology Lausanne, Switzerland
Abdelkader Nasreddine Belkacem, United Arab Emirates University, United Arab Emirates

****Correspondence:***
Claudio Castellini
claudio.castellini@fau.de

Received: *31 December 2021*
Accepted: *25 March 2022*
Published: *29 April 2022*

Citation:
Castellini C (2022) Peripheral Nervous System Interfaces: Invasive or Non-Invasive? Front. Neurorobot. 16:846866. doi: 10.3389/fnbot.2022.846866

INTRODUCTION

Rehabilitation robotics, prosthetics, and assistive robotics are a hard business—especially, *controlling* a rehabilitation robot or a prosthesis is difficult, not only because the user needs to learn to use the device and, possibly, re-learn to use his own body, but also because designing, building, and testing the related control system is difficult. How can a person with no hands, or with severely reduced mobility, let the device know-how and when to grasp, stand, or help perform a reaching movement? Whenever the user's residual ability permits it, an appealing solution, consisting of gathering signals from (intent detection, ID) and stimulating (somatosensory feedback, SF), the peripheral nervous system (PNS), in order to build a bidirectional human-machine, interfaces fostering a deep, synergistic connexion between user and machine (Beckerle et al., 2018).

Now, one of the major hurdles in this field is the generally bad quality of the signals one is interested in gathering and using. Signals coming from the PNS of an impaired user are, *per se,* less stable, reliable, and repeatable than those obtained by able-bodied subjects; moreover, signals gathered in controlled conditions, e.g., while the user sits in a laboratory, can differ dramatically from those the same user produces while trying to perform the same actions in real life (Jiang et al., 2012), hampering the translational potential of research. This problem is being tackled by introducing machine learning able to progressively couple with the user over time (Hahne et al., 2015), allowing for direct, continuous, and sustained interaction with him (Nowak et al., 2018) and by shifting the focus from off-line lab testing to early deployment on end-users in the clinics. There is, however, a second major inherent problem: PNS signals are hard to find. Either we try and build a direct connection to the nerves and/or the muscles (invasive approaches) or we listen to their surface manifestation (non-invasive approaches). Invasiveness means better signals at the price of surgery, to various levels of discomfort for the patient; non-invasiveness is user-friendly and won't violate bodily integrity but yields less clear distorted signals. So, we face a trade-off between the quality of the signals and the quality of user life.

As electronics are more and more integrated with the body, however, more acceptable minimally invasive surgery and permanent implants appear, and their safety and reliability grow. So, eventually, tighter *physical* integration of man and machine could become tolerable and desirable. In a recent perspective paper, (Farina et al., 2021) concur that, at least in prosthetics, where PNS signal processing is the main means to achieve control, osseointegration, targeted muscle reinnervation, and implanted sensors and stimulators are majorly way ahead toward high-performance bionic limbs. One of the messages of the paper is that as technology advances and brings us closer to the biological integration of man and machine, invasive or minimally invasive approaches become more and more appealing, to the point of, eventually, overcoming non-invasive ones and becoming standard.

In this short paper, on the other hand, I argue that PNS non-invasive techniques for both ID and SF will still be indeed preferable in the mid-term. Given the technological advancements we are witnessing in the fields of bodily surface sensing and stimulation, the advantages non-invasive techniques enjoy will still constitute an unsurmountable gap at least in the decade to come. I first talk about non-invasive versus invasive techniques in intent detection (the feed-forward path, section Intent Detection), then in somatosensory feedback (the feedback path, section Somatosensory Feedback), and finally I draw some concluding remarks (section Discussion).

INTENT DETECTION

Control of motorised prostheses *via* "neural" signals started in the 50s, when two surface electromyography (sEMG) sensors were used to determine the speed of opening/closing of a mechanical one-degree-of-freedom gripper (Fougner et al., 2012). This has remained the clinical standard till the mid-2000s, when multi-fingered prosthetic hands started to appear, calling for a more refined form of control, relying on classification (machine learning) applied to sEMG. Despite the promising perspective of the idea though, still, today there are only two commercial ID systems based upon the classification of sEMG: the *Complete Control* by Coapt Engineering[1] and the *MyoPlus* by Ottobock[2], thus, relying on 6-8 single sensors and basic classification. This is not surprising; in general, given the huge variety of daily-living situations, the "one-shot" pattern matching (data collection/model building/control) is unfit for the task (Nowak et al., 2018). Moreover, PNS surface methods, to different degrees, all suffer from sensor displacement caused by donning and doffing, short- and mid-term changes in the morphology of the body, variations in skin impedance, and sensor lift-off (Merletti et al., 2011). The reduced number of sensors that can be embedded in a prosthetic socket is usually insufficient to provide a proper view of muscular activity in the residual limb, and alternative approaches such as pressure sensing and ultrasound scanning are still in the academic prototyping phase (Castellini et al., 2014).

More invasive approaches, on the other hand, promise to provide more focussed, clearer signals, and, as the technology of man-machine integration advances, their appeal increases (Farina et al., 2021). PNS invasive techniques entail different degrees of invasiveness, from the injection of miniaturised EMG sensors into the muscles (Weir et al., 2009) to osseointegration (Ortiz-Catalan et al., 2014) and targeted muscle reinnervation (Aszmann et al., 2015); depending on the severity of the impairment then, patients might agree to stand surgery.

Nevertheless, besides the absence of surgery, surface ID approaches rely on technology requiring (almost) no direct interfacing to biological tissue. Improving surface sensing means achieving higher resolution, both in time and in space, and pushing on miniaturisation; and to achieve this, no complex biological interfaces are needed. To me, this means that, for a long time to come, the non-invasive approaches will still maintain a technological advantage over invasive ones. High-density *wearable* sEMG arrays comprising ∼120 sensors sampled at 2 kHz have recently appeared (e.g., the *MuoviPro* by OTBioelettronica[3]), as well as ultraminiaturised ultrasound scanning devices, whose transducer array fits in the size of a coin (Fournelle et al., 2021). Both systems deliver an unprecedented precision, both in time and space, in the detection of residual muscle activity.

To sum up, high-density surface approaches, coupled with "life-long" interactive machine learning, are not limited to the one-shot schema (Castellini et al., 2015; Beckerle et al., 2018) and shall overcome the problems that are typically associated to non-invasiveness (low density and unreliability in time) *at no additional burden for the user*. For this reason, I argue, they will still represent the methods of choice in the short- and mid-term.

SOMATOSENSORY FEEDBACK

Similar remarks hold, in my opinion, as far as SF is concerned. By SF in this specific framework mechanical or electrical signals are provided to the user's somatosensory system, which relates to the status of the rehabilitation device and/or the environment (Beckerle et al., 2018). SF enables *closed-loop human control* of rehab- and assistive devices: through repeated, stable and identifiable patterns of SF stimulation, corresponding to actions of the device, the user increases the sense of agency (being in control) and ownership (being a part of the self) over it, leading to embodiment and—at least, this is the current opinion in the community—reciprocal adaptation and optimal control (Hahne et al., 2015). The goal is to provide a rich set of feelings to the user, both in terms of the type of feeling (touch, pressure, temperature, texture, etc.) and of its intensity, responsiveness, and appropriateness. At the same time, the stimulator and control system must be as small, lightweight, and low-power as possible (Došen et al., 2017).

Given these goals and despite the spectacular advancements appearing in the scientific literature in the past years, invasive SF is still far from being clinically applicable. Although experiments are reported, in which direct electrical stimulation of peripheral nerves has enabled users to discriminate several tens of spatially stable sensations, such implants are mostly temporary (remarkable exceptions to this are reported, e.g., in Petrini et al., 2019; Ortiz-Catalan et al., 2020), and involve complex surgical procedures; also, nociception seems to be a non-negligible associated issue (Davis et al., 2016). Non-invasive SF, on the other hand, has recently evolved from bulky sets of a few vibrotactile/mechano-tactile actuators to hundreds of high-density electrical stimulators. Electro-tactile stimulation (ECS), for example, works by injecting small electrical currents across the skin, thereby not affecting muscle contractions, and maintaining compatibility with sensing techniques for intent detection (e.g., sEMG, see Došen et al., 2017). High-density integrated ECS delivers highly differentiated stimuli, both qualitatively and quantitatively, at essentially no additional psychological burden for the user (Dideriksen et al., 2022; Isaković et al., 2022). In my opinion, any advantage delivered

[1]https://coaptengineering.com

[2]https://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/myo-plus/myo-plus.html

[3]https://www.otbioelettronica.it/component/sppagebuilder/?view=page&id=150

by direct connection to the afferent nerves, e.g., cuff or intrafascicular electrodes [and there is ample evidence for their effectiveness, see for instance (D'Anna et al., 2019; Petrini et al., 2019)], needs to face an inevitable acceptance gap; moreover, here, too, as it was the case for ID, the technologies involved in ECS vs. invasive techniques belong to two different levels of maturity.

DISCUSSION

This work has concentrated on prosthetics; nevertheless, in my opinion, similar remarks apply to rehabilitation robotics—to a lesser extent, indeed, but in an important way. The usage of ID and SF in the management of rehabilitation exoskeletons, exo-suits, and virtual reality environments is still in its infancy, since robotic therapies for, e.g., stroke survivors and patients with SCI are still largely based upon repetitive motions, not initiated by the patients (if not *via* verbal interaction with a therapist). Here, PNS interfacing is a second-class denizen, although progress in this direction is being made (Lobo-Prat et al., 2014; Sullivan et al., 2017), since, as opposed to the case of prosthetics, patients in rehabilitation are supposed to engage in their therapy only for a limited amount of time and already suffer from neural conditions. For these reasons, it seems to me that ID/SF techniques in rehabilitation should be kept even more non-invasive.

Non-invasive techniques for sensing and stimulation in prosthetics have clear advantages over invasive ones; both in terms of their immediate applicability, but also in perspective, since they do not need direct interfacing with the nervous system. This is just my informed opinion, and this paper is not meant as a survey, and, for sure, does not exhaustively cover the field; so, my statement may be challenged or even overturned in a few years. Still, the technological advancement of non-invasive techniques is faster, and I believe that their advantage over invasive techniques will remain significant for at least one decade; they will still be preferable to invasive ones. Of course, as the technology of invasive techniques progresses, too, there could be a point in the future at which they will be as convenient as non-invasive ones. Mixing invasive and non-invasive techniques could, at that point, be an interesting option to take out the best of both worlds, for instance, coupling ultrasound-based ID and SF are given by implanted stimulators (a thorough review, providing a unified view of such techniques, is found in Shokur et al., 2021). A life-long adaptive form of machine learning will be needed, just as it is today, to reach optimal control of any reha- or assistive device. If it is true that more good data is better, it is also true that data from impaired users change in time, whatever the source of data is, and in that case, a strategy to determine which data to use or to emphasise will be needed. True adaptation to the user should provide this possibility.

AUTHOR CONTRIBUTIONS

The author confirms being the sole contributor of this work and has approved it for publication.

FUNDING

This work was partially supported by the DFG (German Research Agency) project *Deep-Hand*, CA1389/1-2.

REFERENCES

Aszmann, O. C., Roche, A. D., Salminger, S., Paternostro-Sluga, T., Herceg, M., Sturma, A., et al. (2015). Bionic reconstruction to restore hand function after brachial plexus injury: a case series of three patients. *Lancet* 385, 2183–2189. doi: 10.1016/S0140-6736(14)61776-1

Beckerle, P., Castellini, C., and Lenggenhager, B. (2018). Robotic interfaces for cognitive psychology and embodiment research: a research roadmap. *Wiley Interdiscip. Rev. Cogn. Sci.* 10, e1486. doi: 10.1002/wcs.1486

Castellini, C., Artemiadis, P., Wininger, M., Ajoudani, A., Alimusaj, M., Bicchi, A., et al. (2014). Proceedings of the first workshop on Peripheral Machine Interfaces: going beyond traditional surface electromyography. *Front. Neurorobotics*. 8, 22. doi: 10.3389/fnbot.2014.00022

Castellini, C., Bongers, R. M., Nowak, M., and van der Sluis, C. K. (2015). Upper-limb prosthetic myocontrol: two recommendations. *Front. Neurosci.* 9, 496. doi: 10.3389/fnins.2015.00496

D'Anna, E., Valle, G., Mazzoni, A., Strauss, I., Iberite, F., Patton, J., et al. (2019). A closed-loop hand prosthesis with simultaneous intraneural tactile and position feedback. *Sci. Robot.* 4, eaau8892. doi: 10.1126/scirobotics.aau8892

Davis, T. S., Wark, H. A., Hutchinson, D. T., Warren, D. J., O'Neill, K., Scheinblum, T., et al. (2016). Restoring motor control and sensory feedback in people with upper extremity amputations using arrays of 96 microelectrodes implanted in the median and ulnar nerves. *J. Neural Eng.* 13, 036001. doi: 10.1088/1741-2560/13/3/036001

Dideriksen, J. L., Mercader, I. U., and Dosen, S. (2022). Task-dependent adaptations in closed-loop motor control based on electrotactile feedback. *IEEE Trans. Human-Machine Syst*. doi: 10.1109/THMS.2021.3134556. [Epub ahead of print].

Došen, S., Marković, M., Štrbac, M., Belić, M., Kojić, V., Bijelić, G., et al. (2017). Multichannel electrotactile feedback with spatial and mixed coding for closed-loop control of grasping force in hand prostheses. *IEEE Trans. Neural Syst. Rehabil. Eng.* 25, 183–195. doi: 10.1109/TNSRE.2016.2550864

Farina, D., Vujaklija, I., Brånemark, R., Bull, A. M., Dietl, H., Graimann, B., et al. (2021). Toward higher-performance bionic limbs for wider clinical use. *Nat. Biomed. Eng.* doi: 10.1038/s41551-021-00732-x

Fougner, A., Stavdahl, Ø., Kyberd, P. J., Losier, Y. G., and Parker, P. A. (2012). Control of upper limb prostheses: terminology and proportional myoelectric control - a review. *IEEE Trans. Neur. Syst. Rehab. Eng.* 20, 663–677. doi: 10.1109/TNSRE.2012.2196711

Fournelle, M., Grün, T., Speicher, D., Weber, S., Yilmaz, M., Schoeb, D., et al. (2021). Portable ultrasound research system for use in automated bladder monitoring with machine-learning-based segmentation. *Sensors* 21, 6481. doi: 10.3390/s21196481

Hahne, J. M., Dähne, S., Hwang, H.-J., Müller, K.-R., and Parra, L. C. (2015). Concurrent adaptation of human and machine improves simultaneous and proportional myoelectric control. *IEEE Trans. Neur. Syst. Rehabil. Eng.* 23, 618–627. doi: 10.1109/TNSRE.2015.2401134

Isaković, M., Malesević, J., Kostić, M., Došen, S., and Štrbac, M. (2022). The impact of size and position of reference electrode on the localization of biphasic electrotactile stimulation on the fingertips. *IEEE Trans. Haptics.* doi: 10.1109/TOH.2022.3141187. [Epub ahead of print].

Jiang, N., Dosen, S., Müller, K.-R., and Farina, D. (2012). Myoelectric control of artificial limbs - is there a need to change focus? *IEEE Signal Process. Mag.* 29, 148–152. doi: 10.1109/MSP.2012.2203480

Lobo-Prat, J., Kooren, P., Stienen, A., Herder, J., Koopman, B., and Veltink, P. (2014). Non-invasive control interfaces for intention detection in active movement-assistive devices. *J. NeuroEng. Rehabil.* 11, 168. doi: 10.1186/1743-0003-11-168

Merletti, R., Botter, A., Cescon, C., Minetto, M. A., and Vieira, T. M. (2011). Advances in surface EMG: recent progress in clinical research applications. *Crit. Rev. Biomed. Eng.* 38, 347–379. doi: 10.1615/CritRevBiomedEng.v38.i4.20

Nowak, M., Castellini, C., and Massironi, C. (2018). Applying Radical Constructivism to machine learning: a pilot study in assistive robotics. *Constructivist Found.* 13, 250–262. Available online at: https://constructivist.info/13/2/250.nowak

Ortiz-Catalan, M., Håkansson, B., and Brånemark, R. (2014). An osseointegrated human-machine gateway for long-term sensory feedback and motor control of artificial limbs. *Sci. Transl. Med.* 6, 257re6. doi: 10.1126/scitranslmed.3008933

Ortiz-Catalan, M., Mastinu, E., Sassu, P., Aszmann, O., and Brånemark, R. (2020). Self-contained neuromusculoskeletal arm prostheses. *N. Engl. J. Med.* 382, 1732–1738. doi: 10.1056/NEJMoa1917537

Petrini, F. M., Valle, G., Strauss, I., Granata, G., Di Iorio, R., D'Anna, E., et al. (2019). Six-Month assessment of a hand prosthesis with intraneural tactile feedback. *Ann. Neurol.* 85, 137–154. doi: 10.1002/ana.25384

Shokur, S., Mazzoni, A., Schiavone, G., Weber, D. J., and Micera, S. (2021). A modular strategy for next-generation upper-limb sensory-motor neuroprostheses. *Med* 2, 912–937. doi: 10.1016/j.medj.2021.05.002

Sullivan, J. L., Bhagat, N. A., Yozbatiran, N., Paranjape, R., Losey, C. G., Grossman, R. G., et al. (2017). "Improving robotic stroke rehabilitation by incorporating neural intent detection: Preliminary results from a clinical trial," in *2017 International Conference on Rehabilitation Robotics (ICORR)* (London), 122–127. doi: 10.1109/ICORR.2017.8009233

Weir, R. F., Troyk, P. R., DeMichele, G. A., Kerns, D. A., Schorsch, J. F., and Maas, H. (2009). Implantable Myoelectric Sensors (IMESs) for intramuscular electromyogram recording. *IEEE Trans. Biomed. Eng.* 56, 159–171. doi: 10.1109/TBME.2008.2005942

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

frontiers | Frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 02 May 2022
doi: 10.3389/fnbot.2022.883562

Deep Reinforcement Learning Based Trajectory Planning Under Uncertain Constraints

Lienhung Chen[1], *Zhongliang Jiang*[1], *Long Cheng*[2], *Alois C. Knoll*[1] and *Mingchuan Zhou*[1,3*]

[1] Department of Computer Science, Technische Universität München, Munich, Germany, [2] College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou, China, [3] College of Biosystems Engineering and Food Science, Zhejiang University, Hangzhou, China

With the advance in algorithms, deep reinforcement learning (DRL) offers solutions to trajectory planning under uncertain environments. Different from traditional trajectory planning which requires lots of effort to tackle complicated high-dimensional problems, the recently proposed DRL enables the robot manipulator to autonomously learn and discover optimal trajectory planning by interacting with the environment. In this article, we present state-of-the-art DRL-based collision-avoidance trajectory planning for uncertain environments such as a safe human coexistent environment. Since the robot manipulator operates in high dimensional continuous state-action spaces, model-free, policy gradient-based soft actor-critic (SAC), and deep deterministic policy gradient (DDPG) framework are adapted to our scenario for comparison. In order to assess our proposal, we simulate a 7-DOF Panda (Franka Emika) robot manipulator in the PyBullet physics engine and then evaluate its trajectory planning with reward, loss, safe rate, and accuracy. Finally, our final report shows the effectiveness of state-of-the-art DRL algorithms for trajectory planning under uncertain environments with zero collision after 5,000 episodes of training.

Keywords: reinforcement learning, neural networks, trajectory planning, collision avoidance, uncertain environment, robotics

Edited by:
Hang Su,
Fondazione Politecnico di Milano, Italy

Reviewed by:
Bo Lu,
Soochow University, China
Dandan Zhang,
Imperial College London, United Kingdom
Chenming Wu,
Tencent Holdings Limited, China

****Correspondence:***
Mingchuan Zhou
mczhou@zju.edu.cn

Received: *25 February 2022*
Accepted: *28 March 2022*
Published: *02 May 2022*

Citation:
Chen L, Jiang Z, Cheng L, Knoll AC and Zhou M (2022) Deep Reinforcement Learning Based Trajectory Planning Under Uncertain Constraints. Front. Neurorobot. 16:883562. doi: 10.3389/fnbot.2022.883562

1. INTRODUCTION

Multi-Degree-of-Freedom (Multi-DOF) robotic arm is widely used in a variety of automation scenarios, including the automotive industry, equipment fabrication, food industry, health care, and agriculture. In the past, Multi-DOF robotic arms usually operated in isolated, structured environments, and tasks that need to adapt to actual conditions are often done by humans. Human-Robot Collaboration (HRC) combines the flexibility of humans and the efficiency of robots, making manufacturing more flexible and productive (Vysocky and Novak, 2016). However, it is a challenge for traditional motion planning algorithms to define a safe, collision-free HRC system, since all its parameters are established based on a specific environment which makes it difficult to adapt new workspace. Probability Road Map (PRM) and Rapidly-exploring Random Tree (RRT) for instance, are not suitable for dynamics environments, since they require higher real-time performance of algorithms to deal with dynamic obstacles, i.e., they need to construct a real-time mapping of obstacles in the configuration space so as to plan a collision-free path, which is very computationally expensive (Adiyatov and Varol, 2017; Kurosu et al., 2017; Wei and Ren, 2018; Wittmann et al., 2020; Jiang et al., 2021; Liu et al., 2021). Another common approach, potential field (PF), has less computation and better real-time control compared to PRM and RRT, however, it often gets stuck in the local minimum, and has limited performance when the obstacles are in the vicinity of the target (Flacco et al., 2012; Lu et al., 2018, 2021; Xu et al., 2018; Melchiorre et al., 2019; Zhou et al., 2019). Therefore, finding an efficient, safe, and flexible motion planning algorithm is required. Reinforcement learning (RL) paves an alternative way to solve these challenges, especially in many high-dimensional tasks or games, RL can exhibit its outperformance. In DeepMind, it is even thought to be enough to reach general AI (Shanahan et al., 2020).

Recently, deep RL, which leverages neural networks as function approximation, has been proven its effectiveness in many different kinds of high complexity of robotic control tasks. Joshi et al. (2020) shows multiple RGB images with double-deep Q-learning can reach over 80% success rate in different grasping tasks without training on a large dataset. In Gu et al. (2017), the robot learns to complete a door opening task with DDPG and Normalized Advantage Function algorithm (NAF) with only a few hours of training. Another example shown in Haarnoja et al. (2018a) with soft Q-learning a robot can learn how to stack Lego blocks together within 2 h of policy training. Therefore, we carry out an idea, using the DRL-based method to tackle complex trajectory planning under uncertain environments. However, deep RL still faces some challenges: (1) defining an appropriate reward function is not straightforward, especially dealing with high dimensional problems, it is easy to obtain the result we incentivize instead of what we intended. (2) In simple tasks, normally an RL agent can discover an optimal policy in a short period, however when encountering complex tasks it may take a few million training steps to achieve the desired result. (3) It is hard to prevent an RL agent from overfitting, to overcome this problem an agent should be trained on a large distribution of environments, but it's very computationally expensive.

In this article, collision-free trajectory planning under uncertain environments is tackled with state-of-the-art DRL algorithms. Since the robotic systems are high dimensional and the state, action space is continuous, model-free Deep Neural Networks (DNN) approaches for Q- and policy-function approximations are used, which has shown its effectiveness in Amarjyoti (2017). Moreover, we expect our approaches to be more suitable for continuous and stochastic environments as well as to have higher sample efficiency and stability, we leverage a combined version, actor-critic based network, which updates the policy network for better action choices also updates the value network for more precise evaluation on policy at each step (Sutton and Barto, 2018). The primary contributions of this paper are summarized as follows:

- Construct an appropriate dense reward function that includes distance-to-goal reward and distance between obstacles reward, and the weight between both rewards is tuned by comparing the performance across different random seeds, in order to make sure the robot manipulator can follow the goal as long as possible, while also avoid collision with dynamic obstacles.
- Build an uncertain environment in a physics engine to simulate a human coexistent environment and apply state-of-the-art DRL algorithms to 7-DOF robots. Then compare the accuracy, safe rate, and reward of two model-free, policy gradient-based algorithms, SAC, and DDPG.
- To further improve learning efficiency and stability, the state space of goal and obstacles are set to relative position and velocity instead of absolute so that the RL agent can learn the correlation between the end-effector and obstacles as well as the goal, as shown in Section 4.

The rest of the article is structured as follows. Section 2 discusses the study related to the traditional trajectory planning method. Section 3 presents our method and its workflow. Section 4 demonstrates our experiment setup and evaluation of our proposed approaches. Section 5 gives the conclusion of this article and future study.

2. RELATED STUDY

There are currently some possible solutions to trajectory planning and obstacle avoidance. With its probability completeness and exploration efficiency, the RRT has been widely applied in Multi-DOF manipulator's collision-free trajectory planning. Adiyatov and Varol (2017) introduced RRT Fixed Nodes Dynamic (RRT*FND), with the procedures of Reconnect and Regrow in the RRT*FND algorithm, the manipulator can repair the path with an average of 300 ms when encountering an invalid path caused by a dynamic obstacle. Wei and Ren (2018) proposed an improved RRT algorithm, called Smoothly RRT (S-RRT), to generate a smoother path and more stable motion when avoiding obstacles which have shown better exploring speed and exploring efficiency than Basic-RRT and Bi-RRT. In a dual-arm robot pick-and-place environment, Kurosu et al. (2017) regard one of the robot arms as a dynamic obstacle, leveraging the RRT algorithm to effectively avoid collision with another arm during pick-and-place tasks. Although RRT-based algorithm has shown its robustness in either dynamic or static obstacles avoidance, constantly, and randomly moving obstacles avoidance, still requires more research.

Another common approach for collision avoidance trajectory planning is the artificial potential field (APF). This method leverages the PF force of attraction for reaching the goal and repulsion for avoiding obstacles. Xu et al. (2018) leverage a similar algorithm to APF, called velocity potential field (VPF), to avoid collision with a static/dynamic obstacle and a collaborative robot arm. They use the velocity of the robot instead of the distance in APF to avoid suffering from local minima problems when attractive and repulsive forces/velocities confront each other on the same line. Flacco et al. (2012) leverage a simple version of APF, a repulsive vector, generated by the distance to estimate obstacles velocity for collision avoidance. Melchiorre et al. (2019) also leverage the repulsive vector with the distance calculated from the point cloud and have also shown its effectiveness in avoiding collision with static/dynamic obstacles. However, the PF has limited performance when encountering two obstacles that are placed too close to each other. For example, if the goal is in-between or behind two close obstacles, the robot will neglect the goal and turn away.

The other approach is PRM, which takes random samples from the configuration space of the robot and finds a collision-free path between the start and goal nodes. Liu et al. (2021) proposed a Grid-Local PRM that combined a mapping model, sampling strategies, lazy collision detection, and a single local detection method. This proposed method can implement for dynamic path planning for static/dynamic obstacle avoidance. Wittmann et al. (2020) introduce the Obstacle-related Sampling Rejection Probabilistic Roadmap planner (ORSR-PRM). They leverage PRM for trajectory planning and PF for obstacle avoidance in real-time. Similar to the PF method, the probability of generating nodes in between narrow passages is very small, and hence, no path will be planned through the gap.

To overcome the above problem in traditional path planning methods, we proposed another method. Instead of finding a path in configuration space or tackling complicated optimization problems, we leverage the model-free DRL method that allowed the manipulator to autonomously learn optimal collision-free trajectory planning in an uncertain environment.

3. METHODS

The four essential parts of RL are policy, reward function, value function, and model of the environment. With the idea that an intelligent agent should learn to take a sequence of actions that will lead to maximizing cumulative rewards interacting with the environment. Hence, the agent should *exploit* what it has experienced in order to obtain rewards, but also *explore* in order to make better action decisions in the future (Sutton and Barto, 2018). The Basic RL problem is modeled as the Markov decision process (MDP) with elements S_t, A_t, $P(S_{t+1}|S_t, A_t)$, γ, $R(S_{t+1}|S_t, A_t)$, where t represents timestep, S_t and S_{t+1} represent the current state and next state, respectively, A_t stands for the current action, $P(S_{t+1}|S_t, A_t)$ stands for the transition probability of being in S_{t+1} when taking action A_t in the current state S_t, and $\gamma \in [0, 1)$ represents discount factor which determines the importance of future rewards, $R(S_{t+1}|S_t, A_t)$ represents the immediate reward received after transitioning from the current state S_t to the next state S_{t+1}, due to the taken action A_t. In MDP, we assume that the transition probability (or the probability of moving to the next state S_{t+1}) depends on the current state S_t and the decision action A_t. But given S_t and A_t, it is conditionally independent of all previous states and actions.

3.1. Deep Reinforcement Learning

In the case of complex systems such as robotic systems, the explicit model of the dynamics in the environment associated with MDP, i.e., transition probability function, is often not available or difficult to define. Therefore, a model-free-based method is required. Q-learning is one of the most important breakthroughs in RL also known as the off-policy Temporal Difference (TD) control algorithm, defined by

$$\begin{aligned} Q(S_t, A_t) \leftarrow\ & Q(S_t, A_t) \\ & + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)], \end{aligned} \tag{1}$$

where S_t and A_t are state and action in timestep t, respectively. α stands for learning rate. R_{t+1} is the obtained immediate reward due to the taken action A_t. a represents the action that has a maximum Q-value from the state S_{t+1}. The optimal action-value function can be directly approximated by the learned action-value function Q, which dramatically simplified the analysis of the algorithm and has been proven for convergence (Sutton and Barto, 2018).

In traditional Q-learning, we utilize Q-table to help track states, actions, and corresponding expected rewards. However, for continuous action and state-space such as robotic systems, it is infeasible to build up a large table. Therefore, we need a function approximation for the action-value function Q and a DNN is one of the efficient and easy techniques to approximate a non-linear function. However, RL with DNN is pretty unstable, the weights of the network can oscillate or diverge due to the high correlation between actions and states. To overcome this issue, we need to leverage two important techniques, *Experience Replay*, and *Target Network*. By Experience Replay, the agent's experience at each time step will be stored in replay memory as the tuple $(S_t, A_t, R_{t+1}, S_{t+1})$, and when the replay memory size is equal to or bigger than a mini-batch size, we then uniformly sample the memory randomly for a mini-batch of experience and use this to learn off-policy, in order to break the correlation (Lin, 1992). Moreover, to make training more stable, a target network is used for calculating the estimate of optimal future value $\max_a Q(S_{t+1}, a)$ in the Bellman equation, and hence, the loss function can be defined as

$$\begin{aligned} L(\theta) = \{ & [R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_{target})] \\ & - Q(S_t, A_t; \theta_{prediction})\}^2, \end{aligned} \tag{2}$$

where $\theta_{prediction}$ are prediction network's weights updated in every iteration, whereas θ_{target} are the target network's weights, which are not trained but periodically synchronized with the parameters of the prediction Q-network.

3.2. The Proposed DRL-Based Trajectory Planning for Uncertain Environments

In this section, we define the setup for the DRL framework, such as the state space $\mathcal{S}$, the action space $\mathcal{A}$, and the reward function.

3.2.1. State Space

In our experiment the robot manipulator, we used is 7-DOF, therefore, if we set joint positions and velocity as the observations, the learning efficiency of the agent is quite low (Henderson et al., 2018) or may even be unable to find the optimal trajectory. To overcome this issue, we instead use the end-effector position p_e and velocity $\dot{p_e}$ then calculate inverse kinematic (IK) to control joint position. Moreover, we use relative position and velocity to the end-effector instead of obstacles or

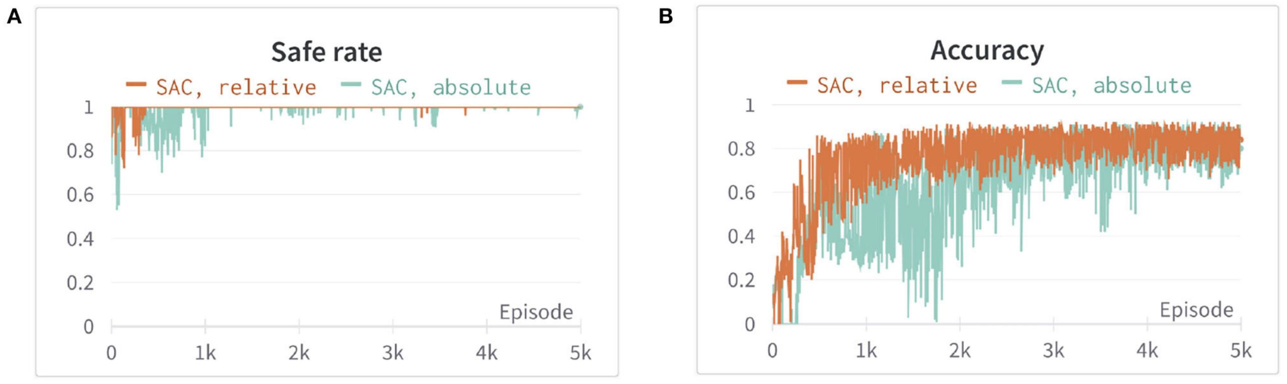


FIGURE 1 | The comparison of two different observation spaces set up in the first environment. Both are using soft actor-critic (SAC) and with the same hyperparameters setting. The orange line is the result using relative position and velocity in observation space, whereas the blue line is using position and velocity. **(A)** The safe rate (defined in Section 4.2) of different observation spaces. **(B)** The accuracy (defined in Section 4.2) of different observation spaces. Each episode corresponds to 100 time steps.

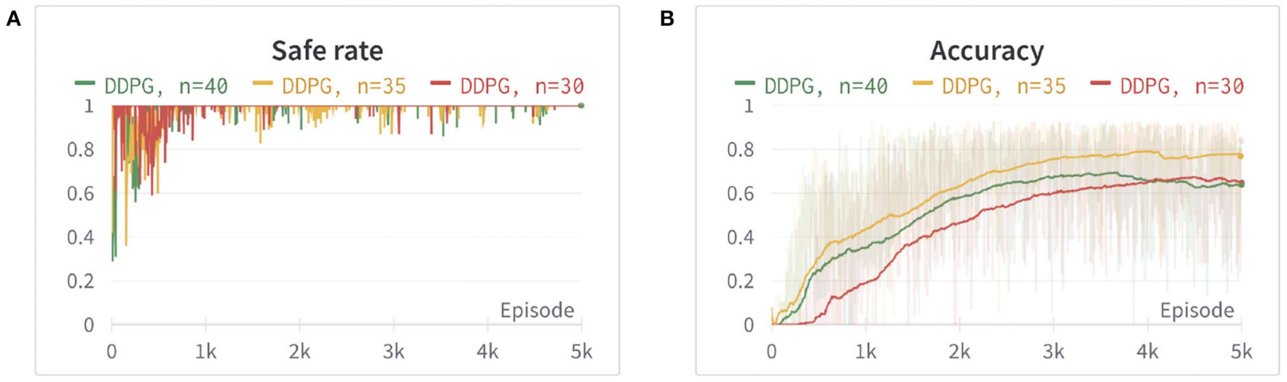


FIGURE 2 | The comparison between different power (n) of the exponential decay function. **(A)** The safe rate (defined in Section 4.2) of three different power (n) of the exponential decay function. **(B)** The accuracy (defined in Section 4.2) of three different power (n) of the exponential decay using an exponential moving average for better visualization. The two figures show that $n = 35$ has better learning efficiency, safe rate, and accuracy.

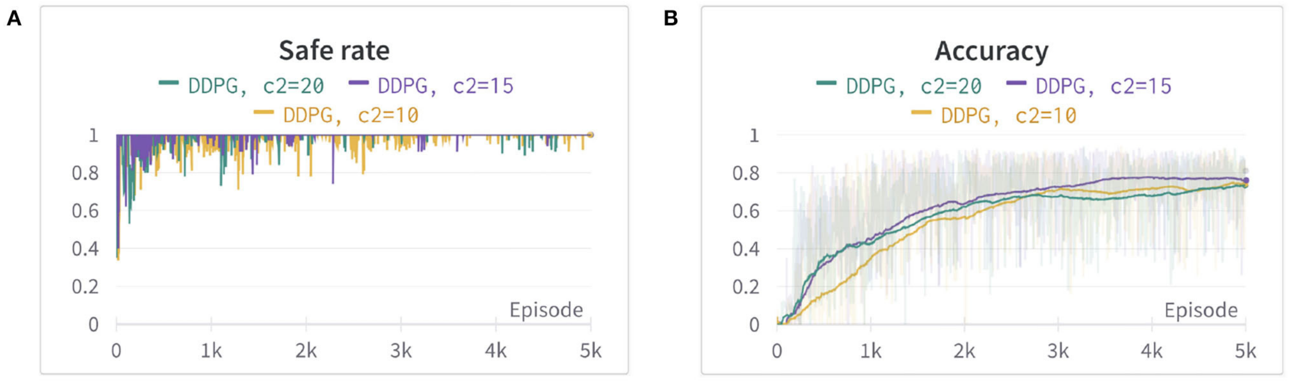


FIGURE 3 | The comparison between different weights of reward R_O. **(A)** The safe rate (defined in Section 4.2) of three different weights of reward R_O. **(B)** The accuracy (defined in Section 4.2) of three different weights of reward R_O using an exponential moving average for better visualization. The two figures show that $c2 = 15$ has better learning efficiency, safe rate, and accuracy.

the goal position $\bar{p_o}/\bar{p_t}$ and velocity $\dot{\bar{p_o}}$ / $\dot{\bar{p_t}}$, which has shown faster convergence and higher stability in **Figure 1**. The above information is assumed known and obtained from the sensor. Furthermore, to increase learning efficiency, we constrain the manipulator in a specific workspace, and hence, the robot will only explore its reachable area and the area with the goal nearby. The State-space S is hence defined as

$$\mathcal{S} = \{p_e,\ \dot{p_e},\ \bar{p_t},\ \dot{\bar{p_t}},\ \bar{p_o},\ \dot{\bar{p_o}}\}. \tag{3}$$

3.2.2. Action Space

As we mentioned in section A, we set the end-effector position as the observation for better learning efficiency. Therefore, we can reduce the dimension of actions from seven dimensions to three dimensions with fixed orientations. The action space $\mathcal{A}$ is defined as

$$\mathcal{A} = \{\Delta x,\ \Delta y,\ \Delta z\}, \tag{4}$$

where Δx,Δy,Δz are bounded between -0.1 and 0.1 such that we can avoid sudden movements of the robotic arm in every time step due to excessive output of the action.

3.2.3. Reward Function

The reward function mixes reward variables into a single output value and provides feedback for an agent to learn what we incentive. In our case, we expect the robot to follow the goal as long as possible while avoiding dynamic obstacles. Therefore, we define the reward function by a weighted sum of two terms: First, the distance between the end-effector and the goal. Second, the closest distance between the robot manipulator and

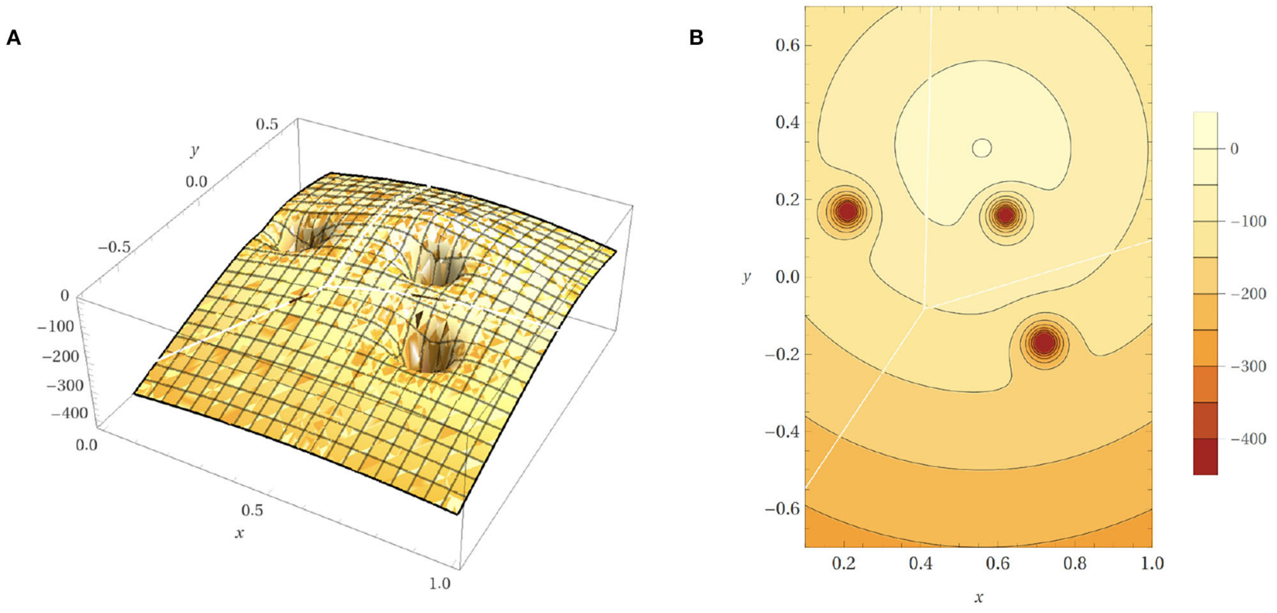


FIGURE 4 | Reward function on the planar section of the workspace. **(A)** The 3D plot of the reward function. **(B)** Contour plot of reward function.

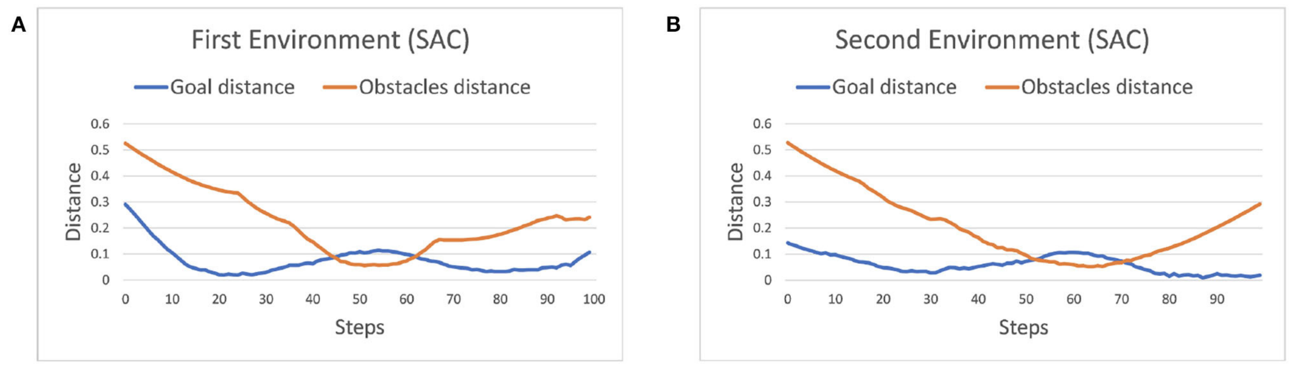


FIGURE 5 | The behavior of the manipulator with respect to the distance(m) in one of the episodes (100-time steps). **(A)** Distance between robot and obstacles as well as the target (first scenario). **(B)** Distance between robot and obstacles as well as the target (second scenario).

obstacles. Moreover, we use negative a reward over a positive so that the robot will try its best to avoid penalties and, hence, can learn as quickly as possible. The reward function is defined as

$$R = -c_1 R_T - c_2 R_O, \tag{5}$$

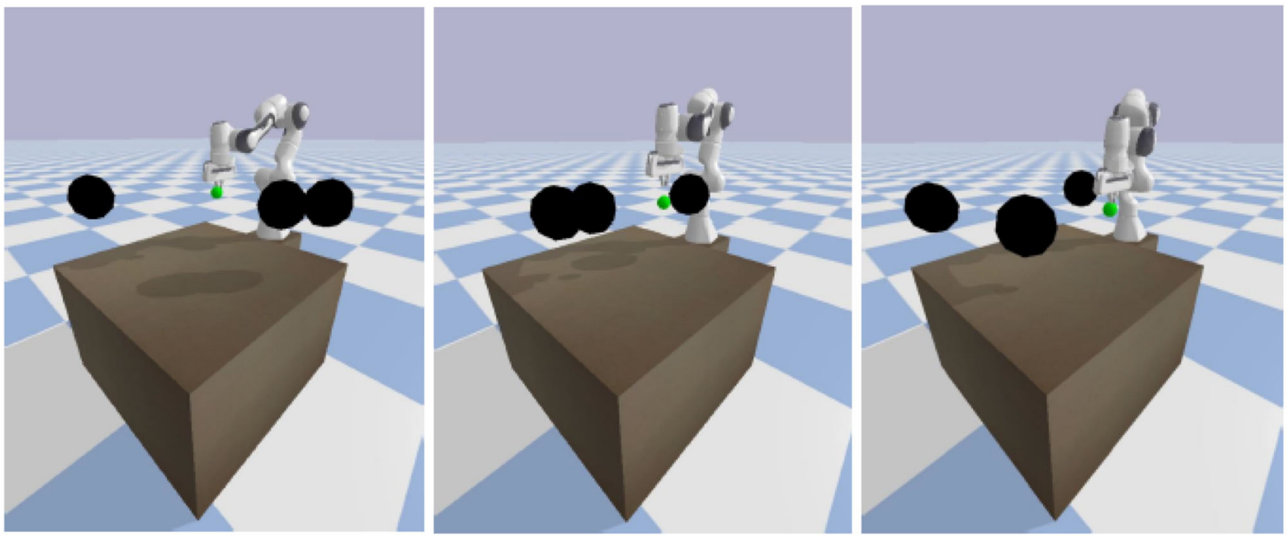
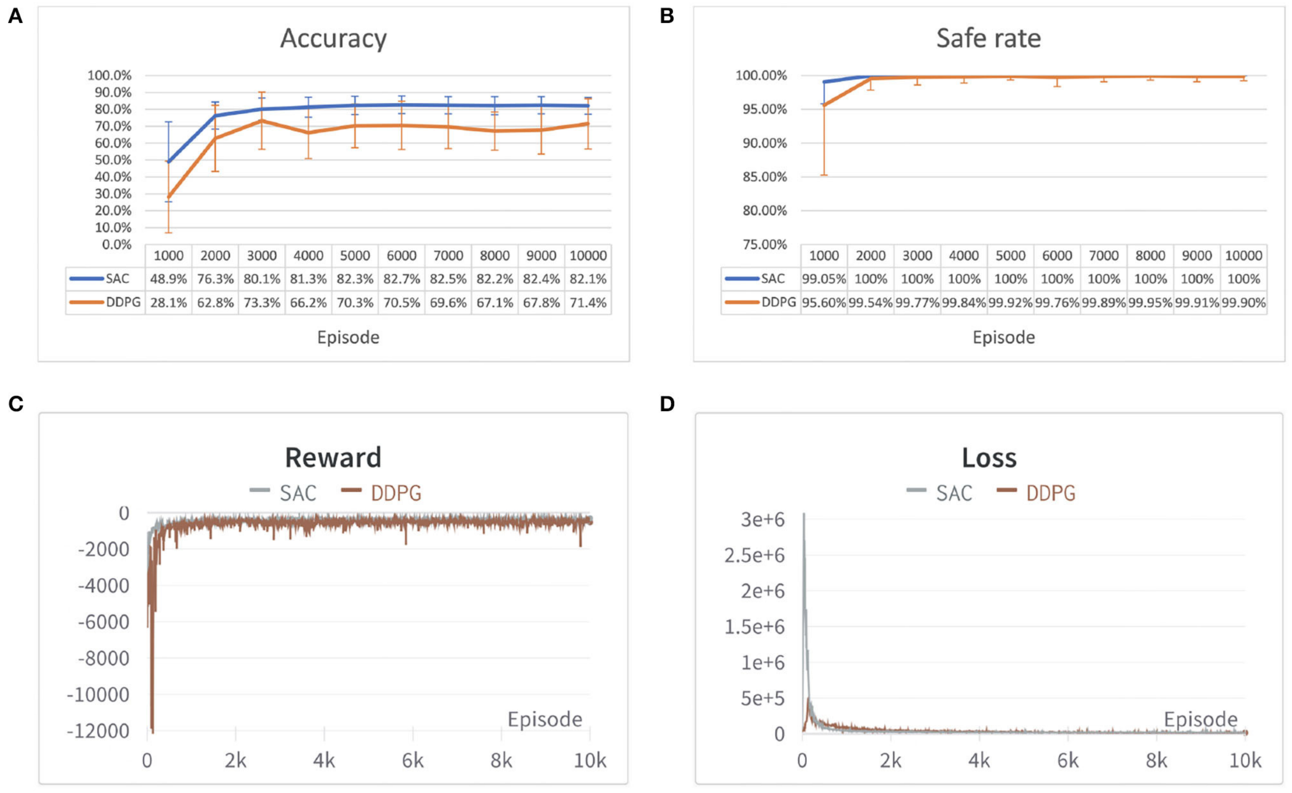


FIGURE 6 | From left to right, first, the robot learns to reach the goal. Second, avoid collisions with dynamic obstacles. Third, keep reaching the goal.



	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
SAC	48.9%	76.3%	80.1%	81.3%	82.3%	82.7%	82.5%	82.2%	82.4%	82.1%
DDPG	28.1%	62.8%	73.3%	66.2%	70.3%	70.5%	69.6%	67.1%	67.8%	71.4%

	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
SAC	99.05%	100%	100%	100%	100%	100%	100%	100%	100%	100%
DDPG	95.60%	99.54%	99.77%	99.84%	99.92%	99.76%	99.89%	99.95%	99.91%	99.90%

FIGURE 7 | Performance comparison of SAC and Deep Deterministic Policy Gradient (DDPG) algorithm in the first environment. **(A)** Accuracy of different algorithms shown in error bar line graph. **(B)** A safe rate of different algorithms in the error bar line graph. **(C)** The cumulative reward for each episode. **(D)** Loss for each episode. Each episode corresponds to 100 time steps.

where R_T is the reward obtained from the distance between the end-effector and the goal using Euclidean distance (m),

$$R_T = \frac{1}{2} d_T^2, \tag{6}$$

where d_T is the Euclidean distance between the end-effector and the goal. The reward R_O is obtained from the closest distance (m) between the robot manipulator and obstacles,

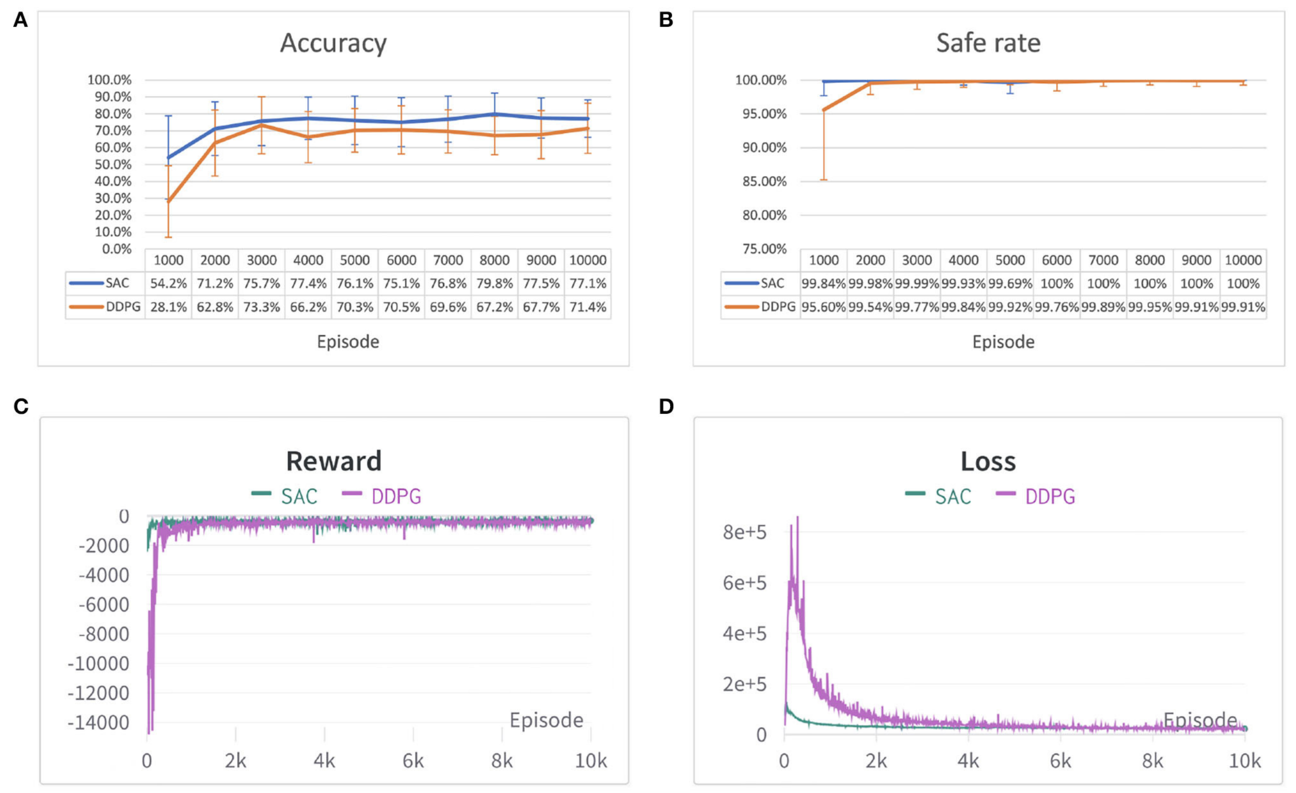
$$R_O = \left(\frac{a}{a + d_O}\right)^n, \tag{7}$$

where d_O is the closest distance from the obstacles computed by PyBullet. a is set to 1, in order to avoid the denominator equaling zero when a collision happens. The power of exponential decay function $n = 35$ and the weights $c2 = 15$ are determined by using trial and error. We set $c1$ as a fixed value of 500 and tune the parameters n and $c2$ by evaluating the safe rate, accuracy, and learning efficiency, as shown in **Figures 2**, **3** (Since DDPG is more sensitive to parameters, we use DDPG for comparison; Haarnoja et al., 2018b). In order to show our reward function has a maximum, we plot our reward function on the planar section of the workspace, as shown in **Figure 4**. Moreover, since the dynamic/static goal and dynamic obstacles are on the same x-y plane, it can be demonstrated in 3D space instead of 4D for better visualization. As it can be observed from **Figure 4**, the reward decreases as the robot's end-effector moves toward obstacles and increases as it moves toward the goal, and when the end-effector reaches the goal point, the reward is maximum. The behavior of the robot manipulator in two environments is also shown in **Figure 5**. The distance between the end-effector and goal diminishes as the robot approaches the goal and when obstacles are close to the body of the robot, the robot backs off until obstacles move away from the manipulator.

3.2.4. Deep Deterministic Policy Gradient

Deep deterministic policy gradient (DDPG), introduced in Lillicrap et al. (2015), is an actor-critic, model-free algorithm based on the deterministic policy gradient that can operate over continuous action spaces. Traditionally, in policy gradient-based algorithms the policy function is always stochastic, i.e., it is modeled as a probability distribution over actions given the current state. In DDPG, the policy function is instead modeled as a deterministic decision. However, this may lead to a low exploration issue, and hence, they add additive noise to the deterministic action to explore the environment, which is represented as:

$$\mu'(S_t) = \mu(S_t|\theta_t^{\mu}) + \mathcal{N}, \tag{8}$$



	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
SAC	54.2%	71.2%	75.7%	77.4%	76.1%	75.1%	76.8%	79.8%	77.5%	77.1%
DDPG	28.1%	62.8%	73.3%	66.2%	70.3%	70.5%	69.6%	67.2%	67.7%	71.4%

Episode

	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
SAC	99.84%	99.98%	99.99%	99.93%	99.69%	100%	100%	100%	100%	100%
DDPG	95.60%	99.54%	99.77%	99.84%	99.92%	99.76%	99.89%	99.95%	99.91%	99.91%

Episode

FIGURE 8 | Performance comparison of SAC and DDPG algorithm in the second environment. **(A)** Accuracy of different algorithms shown in error bar line graph. **(B)** A safe rate of different algorithms in the error bar line graph. **(C)** The cumulative reward for each episode. **(D)** Loss for each episode. Each episode corresponds to 100 time steps.

Algorithm 1 | DDPG-based trajectory planning.

```
Input: batch size: $B$, target smoothing coefficient $\tau$, discount factor: $\gamma$, number of training episode: $M$, timesteps of each episode: $T$
Randomly initialize Q network $Q(S,A|\theta^Q)$ and policy network $\mu(S|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$
Initialize target Q network $Q'$ and target policy network $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $D \leftarrow \phi$
Input: A set of observation state $\mathcal{S} = \{p_e, \dot{p}_e, \bar{p}_t, \dot{\bar{p}}_t, \bar{p}_o, \dot{\bar{p}}_o\}$
Output: A set of optimal policy $\mathcal{A} = \{\Delta x, \Delta y, \Delta z\}$
  for episode = 1, $M$ do
      Initialize a random noise $\mathcal{N}$ for action exploration
      Receive initial observation state $S_1$
      for t = 1, $T$ do
          Select action $A_t = \mu(S_t|\theta^\mu) + \mathcal{N}_t$ according to the
             current policy and exploration noise
          Execute action $A_t$ and observe reward $R_t$ and
             observe new state $S_{t+1}$
          Store transition $(S_t, A_t, R_t, S_{t+1})$ in $D$
          Sample a random minibatch of $B$ transitions
             $(S_i, A_i, R_i, S_{i+1})$ from $D$
          Set $y_i = R_i + \gamma Q'[S_{i+1}, \mu'(S_{i+1}|\theta^{\mu'})|\theta^{Q'}]$
          Update critic by minimizing the loss:
             $L = \frac{1}{B}\sum_i [y_i - Q(S_i, A_i|\theta^Q)]^2$
          Update the actor policy using the sampled policy
             gradient: $\nabla_{\theta^\mu} J \approx$
             $\frac{1}{B}\sum_i \nabla_A Q(S,A|\theta^Q)|_{S=S_i, A=\mu(S_i)} \nabla_{\theta^\mu}\mu(S|\theta^\mu)|_{S_i}$
          Update the target networks:
             $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
             $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
      end for
  end for
```

where $\mu(S_t)'$ and $\mathcal{N}$ are the exploration policy and additive noise, here, use an Ornstein-Uhlenbeck process. $\mu(S_t|\theta_t^\mu)$ and θ_t^μ are the output action and parameters of the actor-network. Moreover, the traditional target networks are updated with the parameters of the trained networks every couple of thousand steps, which may cause big differences between the two updates. Therefore, they introduced a *soft target update*, which is actually better to make the target networks slowly track the trained networks, by updating their parameters after each update of the trained network using a sliding average for both the actor and the critic:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta', \; with\; \tau \ll 1, \tag{9}$$

where θ and θ' represent parameters for the actor- or critic-network and the target actor- or target critic-network. τ is the target smoothing coefficient. The Experience Replay mentioned in Section A is also used here to store past trajectories and provides samples of them to perform gradient updates for better learning efficiency. The detailed pseudo algorithm of DDPG-based trajectory planning is shown in **Algorithm 1**.

3.2.5. Soft Actor-Critic

Similar to DDPG, soft actor-critic (SAC) introduced in Haarnoja et al. (2018b), is also an actor-critic, model-free algorithm that can operate over continuous action spaces, but is based on the stochastic policy by maximizing the expected reward of the actor while maximizing entropy, i.e., achieve the goal while acting as randomly as possible. Hence, the general maximum entropy can be represented as:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(S_t,A_t)\sim\rho_\pi}[r(S_t,A_t) + \alpha\mathcal{H}(\pi(\cdot|S_t))], \tag{10}$$

where ρ_π and $\mathcal{H}(\pi(\cdot|s_t))$ are the policy and the entropy. α is the temperature parameter for determining the relative importance of the entropy term against the reward, and hence controls the stochasticity of the optimal policy. Since SAC is an off-policy algorithm, the Experience Replay is also used to improve the learning efficiency. Moreover, in order to decrease the changes between two updates, the soft target update technique in Equation (4) is also applied. In Haarnoja et al. (2018b), they also compare with some other on-policy DRL algorithms, such as TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017), or A3C (Mnih et al., 2016), and has shown its higher sample efficiency. Compare with DDPG, it also has lower hyperparameter sensitivity and higher stability, which is also our case shown in Section 4. The detailed pseudo algorithm of SAC-based trajectory planning is shown in **Algorithm 2**.

4. EXPERIMENT AND RESULTS

In this section, we show that DDPG and SAC can learn optimal trajectory planning for dynamic obstacles collision avoidance. For the evaluation, we compare two different DRL algorithms with safe rate, accuracy, and reward in two different environments.

4.1. Environment Setup

In our experiment, we applied the proposed collision avoidance DDPG and SAC algorithm on a 7-DOF manipulator (Panda from Franka Emika) simulated in a PyBullet physics engine and leveraged the RL toolkit Gym. The environment setup contains a manipulator, a table, a green sphere goal, and three black sphere obstacles, as shown in **Figure 6**. Moreover, we constructed two environments for the evaluation, either a static goal and dynamic obstacles or a dynamic goal and dynamic obstacles. Besides, in order to make sure our model can learn under uncertainty, the starting positions of the goal and three obstacles are uniformly and randomly sampled in a specific range, and the goal moving areas are constrained in the robot's reachable area.

4.2. Evaluation

We evaluate different DRL algorithms in two defined scenarios with the safe rate, accuracy, reward, and loss. Each episode corresponds to 100 time steps, i.e., the robot has to reach the goal and avoid collision within 100 time steps. The safe rate represents

Algorithm 2 | SAC-based trajectory planning.

```
Input: batch size: $B$, target smoothing coefficient $\tau$, discount factor: $\gamma$, number of training episode: $M$, timesteps of each episode: $T$
Randomly initialize Q network $Q_1$, $Q_2$, policy network $\pi$ and value network $V$ with weights $\theta_1, \theta_1, \phi$ and $\psi$.
Initialize target Q networks $Q'_1$ and $Q'_2$, target value network $V'$ with weights $\theta'_1 \leftarrow \theta, \theta'_2 \leftarrow \theta$ and $\psi' \leftarrow \psi$
Initialize replay buffer $D$
Input: A set of observation state $\mathcal{S} = \{p_e, \dot{p_e}, \bar{p_t}, \dot{\bar{p_t}}, \bar{p_o}, \dot{\bar{p_o}}\}$
Output: A set of optimal policy $\mathcal{A} = \{\Delta x, \Delta y, \Delta z\}$
  for episode = 1, $M$ do
      Receive initial observation state $S_1$
      for t = 1, $T$ do
          Select action $A_t \sim \pi_\phi(A_t|S_t)$
          Execute action $A_t$ and observe reward $R_t$ and
             observe new state $S_{t+1}$
          Store transition $(S_t, A_t, R_t, S_{t+1})$ in $D$
          Sample a random minibatch of $B$ transitions
             $(S_i, A_i, R_i, S_{i+1})$ from $D$
          Update V by minimizing the mean squared error:
             $\nabla_\psi J_V(\psi) = \frac{1}{B}\sum_i \nabla_\psi V_\psi(S_i)[V_\psi(S_i)$
             $- \min_{j=1,2} Q_{\theta'_j}(S_i, A_i) + log\pi_\phi(A_i|S_i)]$
          Update Q by minimizing the soft Bellman residual:
             $\nabla_{\theta_{1,2}} J_Q(\theta_{1,2}) = \nabla_{\theta_{1,2}} \frac{1}{B}\sum_i$
             $\{[Q_{\theta_1}(S_i, A_i) - \alpha R(S_i, A_i) - \gamma V_{\psi'}(S_{i+1})]^2 -$
             $[Q_{\theta_2}(S_i, A_i) - \alpha R(S_i, A_i) - \gamma V_{\psi'}(S_{i+1})]^2\}$
          Update $\pi$ by minimizing the expected KL-
             divergence:
             $\nabla_\phi J_\pi(\phi) = \nabla_\phi \frac{1}{B}\sum_i [log\pi_\phi(A_i|S_i) -$
             $\min_{j=1,2} Q_{\theta'_j}(S_i, A_i)]$
          Update the target value networks:
             $\psi' \leftarrow \tau\psi + (1-\tau)\psi'$
      end for
  end for
```

the number of time steps without collision divided by 100 time steps (one episode),

$$Safe\ rate = \frac{number\ of\ timesteps\ without\ collision}{100\ timesteps}. \tag{11}$$

The accuracy stands for a success rate of keeping a distance between the target within 0.05 *m* while far away from obstacles, and 0.12 *m* while avoiding collision with obstacles (the radius of obstacles is 0.1 *m*), therefore, it is not possible to obtain 100% of accuracy, since it also includes time steps from the rest position to the target. The accuracy is set as,

$$Accuracy = \frac{number\ of\ timesteps\ that\ succeed}{100\ timesteps}. \tag{12}$$

The two algorithms' performances were evaluated in two different environments using the same set of parameters respectively, such as learning rate, number of hidden layers, target smoothing coefficient. The environment settings considered for the experiments are (1) dynamic goal and dynamic obstacles: the starting position of both goal and obstacles are uniformly and randomly sampled in a specific range for each episode, and moving with constant speed. (2) fixed goal and dynamic obstacles: the obstacles remain in the same setting as the first one, but with a fixed goal sampled randomly for each episode.

In each experiment, the safe rate, accuracy, reward, and loss per episode have been traced during the training process and compared after 10,000 episodes of 100 time steps each. The result of the two environments is shown in **Figures 7**, **8**. From both **Figures 7C,D**, **8C,D** it can be observed that both algorithms' cumulative reward converges to their maximum value, and the losses do not have significant reductions, i.e., the robot has learned a stable optimal trajectory planning under an uncertain environment. Moreover, both **Figures 7A**, **8A** show that SAC performs much more consistently, efficiently, and higher accuracy, whereas deterministic policy-based DDPG exhibits high variability between episodes and less stable. Furthermore, both **Figures 7B**, **8B** demonstrate that SAC can learn a collision free trajectory with 100% of safe rate within 6,000 episodes, while DDPG still cannot guarantee to reach a 100% of safe rate within 10,000 episodes. Similar results are also corroborated in Gu et al. (2016) and Haarnoja et al. (2018b). The reason for that is because the interplay between the deterministic actor-network and the Q-function makes DDPG unstable and sensitive to hyperparameters, especially for complex and high-dimensional tasks, however, DDPG still shows its effectiveness in both scenarios. Overall, the performance of the stochastic policy-based SAC is more stable and consistent when dealing with complex tasks.

5. CONCLUSION AND FUTURE STUDY

In this article, we presented two state-of-the-art off-policy DRL approaches that can be used to discover optimal trajectory planning under an uncertain environment. Especially, stochastic policy-based SAC can achieve an average of 82% of accuracy in the first scenario and 79% in the second, moreover, with lower variability between episodes and zero collision after 5,000 episodes. The results show the clear potential of the proposed approaches in the application of an uncertain environment, such as HRC scenarios. The future study will transfer the trained model from a simulation environment to real physical robotic manipulators and transfer the learning skill from simulation to the real environment with visual sensing.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

LChen implemented the code and draft the manuscript. ZJ assisted to implement the code and discussed the manuscript. LCheng assisted to implement the code and discussed the manuscript. AK guided the research and discussed the results. MZ guided the research, implemented parts of code, and revised the manuscript.

FUNDING

This study was supported by the National Natural Science Foundation of China (Grant Numbers: 32101626 and 61902442) and ZJU 100 Young Talent Program. This study was supported in part by the German Research Foundation (DFG) and in part by the Technical University of Munich (TUM) in the framework of the Open Access Publishing Program.

REFERENCES

Adiyatov, O., and Varol, H. A. (2017). "A novel RRT*-based algorithm for motion planning in dynamic environments," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)* (Takamatsu), 1416–1421. doi: 10.1109/ICMA.2017.8016024

Amarjyoti, S. (2017). Deep reinforcement learning for robotic manipulation-the state of the art. *arXiv preprint arXiv:1701.08878*. doi: 10.48550/arXiv.1701.08878

Flacco, F., Kröger, T., De Luca, A., and Khatib, O. (2012). "A depth space approach to human-robot collision avoidance," in *2012 IEEE International Conference on Robotics and Automation* (St Paul, MN), 338–345. doi: 10.1109/ICRA.2012.6225245

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (Singapore), 3389–3396. doi: 10.1109/ICRA.2017.7989385

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. (2016). Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*. doi: 10.48550/arXiv.1611.02247

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., and15 Levine, S. (2018a). "Composable deep reinforcement learning for robotic manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6244–6251. doi: 10.1109/ICRA.2018.8460756

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

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

Jiang, Z., Li, Z., Grimm, M., Zhou, M., Esposito, M., Wein, W., et al. (2021). Autonomous robotic screening of tubular structures based only on real-time ultrasound imaging feedback. *IEEE Trans. Indus. Electron*. 69, 7064–7075. doi: 10.1109/TIE.2021.3095787

Joshi, S., Kumra, S., and Sahin, F. (2020). "Robotic grasping using deep reinforcement learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)* (Hong Kong), 1461–1466. doi: 10.1109/CASE48305.2020.9216986

Kurosu, J., Yorozu, A., and Takahashi, M. (2017). Simultaneous dual-arm motion planning for minimizing operation time. *Appl. Sci*. 7:1210. doi: 10.3390/app7121210

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*. doi: 10.48550/arXiv.1509.02971

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn*. 8, 293–321. doi: 10.1007/BF00992699

Liu, Y., Chen, B., Zhang, X., and Li, R. (2021). Research on the dynamic path planning of manipulators based on a grid-local probability road map method. *IEEE Access* 9, 101186–101196. doi: 10.1109/ACCESS.2021.3098044

Lu, B., Chu, H. K., Huang, K., and Cheng, L. (2018). Vision-based surgical suture looping through trajectory planning for wound suturing. *IEEE Trans. Autom. Sci. Eng*. 16, 542–556. doi: 10.1109/TASE.2018.2840532

Lu, B., Li, B., Chen, W., Jin, Y., Zhao, Z., Dou, Q., et al. (2021). Toward image-guided automated suture grasping under complex environments: a learning-enabled and optimization-based holistic framework. *IEEE Trans. Autom. Sci. Eng*. 1–15. doi: 10.1109/TASE.2021.3136185

Melchiorre, M., Scimmi, L. S., Pastorelli, S. P., and Mauro, S. (2019). "Collison avoidance using point cloud data fusion from multiple depth sensors: a practical approach," in *2019 23rd International Conference on Mechatronics Technology (ICMT)* (Salerno), 1–6. doi: 10.1109/ICMECT.2019.8932143

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning* (New York, NY), 1928–1937.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). "Trust region policy optimization," in *International Conference on Machine Learning* (Lille), 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. doi: 10.48550/arXiv.1707.06347

Shanahan, M., Crosby, M., Beyret, B., and Cheke, L. (2020). Artificial intelligence and the common sense of animals. *Trends Cogn. Sci*. 24, 862–872. doi: 10.1016/j.tics.2020.09.002

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA; London: MIT Press.

Vysocky, A., and Novak, P. (2016). Human-robot collaboration in industry. *MM Sci. J*. 9, 903–906. doi: 10.17973/MMSJ.2016_06_201611

Wei, K., and Ren, B. (2018). A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. *Sensors* 18:571. doi: 10.3390/s18020571

Wittmann, J., Jankowski, J., Wahrmann, D., and Rixen, D. J. (2020). "Hierarchical motion planning framework for manipulators in human-centered dynamic environments," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)* (Naples), 525–532. doi: 10.1109/RO-MAN47096.2020.9223549

Xu, X., Hu, Y., Zhai, J., Li, L., and Guo, P. (2018). A novel non-collision trajectory planning algorithm based on velocity potential field for robotic manipulator. *Int. J. Adv. Robot. Syst*. 15:1729881418787075. doi: 10.1177/1729881418787075

Zhou, M., Yu, Q., Huang, K., Mahov, S., Eslami, A., Maier, M., et al. (2019). Towards robotic-assisted subretinal injection: a hybrid parallel-serial robot system design and preliminary evaluation. *IEEE Trans. Indus. Electron*. 67, 6617–6628. doi: 10.1109/TIE.2019.2937041

PERSPECTIVE
published: 10 May 2022
doi: 10.3389/fnbot.2022.886050

ROS-Neuro: An Open-Source Platform for Neurorobotics

Luca Tonin[1,2], Gloria Beraldo[1,3], Stefano Tortora[1,2] and Emanuele Menegatti[1,2]*

[1] Intelligent Autonomous Systems Laboratory, Department of Information Engineering, University of Padova, Padua, Italy, [2] Padova Neuroscience Center, University of Padova, Padua, Italy, [3] Institute of Cognitive Sciences and Technologies, National Research Council, Rome, Italy

The growing interest in neurorobotics has led to a proliferation of heterogeneous neurophysiological-based applications controlling a variety of robotic devices. Although recent years have seen great advances in this technology, the integration between human neural interfaces and robotics is still limited, making evident the necessity of creating a standardized research framework bridging the gap between neuroscience and robotics. This perspective paper presents Robot Operating System (ROS)-Neuro, an open-source framework for neurorobotic applications based on ROS. ROS-Neuro aims to facilitate the software distribution, the repeatability of the experimental results, and support the birth of a new community focused on neuro-driven robotics. In addition, the exploitation of Robot Operating System (ROS) infrastructure guarantees stability, reliability, and robustness, which represent fundamental aspects to enhance the translational impact of this technology. We suggest that ROS-Neuro might be the future development platform for the flourishing of a new generation of neurorobots to promote the rehabilitation, the inclusion, and the independence of people with disabilities in their everyday life.

Keywords: ROS, ROS-Neuro, neural interface, brain-machine interface, neurorobotics

OPEN ACCESS

Edited by:
Florian Röhrbein, Technische Universität Chemnitz, Germany

Reviewed by:
Onofrio Gigliotta, University of Naples Federico II, Italy
Eris Chinellato, Middlesex University, United Kingdom

****Correspondence:***
Luca Tonin
luca.tonin@dei.unipd.it

Received: *28 February 2022*
Accepted: *07 April 2022*
Published: *10 May 2022*

Citation:
Tonin L, Beraldo G, Tortora S and Menegatti E (2022) ROS-Neuro: An Open-Source Platform for Neurorobotics. Front. Neurorobot. 16:886050. doi: 10.3389/fnbot.2022.886050

1. INTRODUCTION

The last few years have seen a growing interest in the topic of neural human-machine interfaces as a novel—potentially groundbreaking—interaction modality between users and robotic devices. In these interfaces, neurophysiological signals are acquired in real-time [e.g., from electroencephalography (EEG) or from electromyography (EMG)], processed with minimum delay, and translated into commands for the external actuators. Based on this workflow, researchers have demonstrated the feasibility and the potentiality of this innovation, in particular for those people suffering from severe motor disabilities (Kennedy and Bakay, 1998; Hochberg et al., 2012; Aflalo et al., 2015; Chaudhary et al., 2016; Tonin and Millán, 2021). For instance, the latest advances in the brain-machine interface (BMI) showed the possibility to exploit brain signals (acquired with invasive or non-invasive techniques) to control telepresence robots, powered wheelchairs, robotic arms, and upper/lower-limb exoskeletons (Iez et al., 2010; Leeb et al., 2013, 2015; Liu et al., 2017, 2018; Edelman et al., 2019). In parallel, systems relying on residual motor functions demonstrated that EMG signals can be re-interpreted and used to precisely drive robotic arms in amputees (Farrell and Weir, 2008; Castellini et al., 2009; Cipriani et al., 2011; Borton et al., 2013; Parajuli et al., 2019), to initiate the walking pattern in lower-limb exoskeletons (Sylos-Labini et al., 2014; De Luca et al., 2019) or to support reaching and grasping tasks with upper-limb exoskeletons (Batzianoulis et al., 2017, 2018; Betti et al., 2018).

However, despite such an emerging and promising trend, the full potential of the field is still unrevealed. Among the multifaceted and multidisciplinary aspects belonging to the neurorobotics challenge, herein we propose an engineering perspective on the development of neural driven robotic devices. In this regard, we highlight three current drawbacks that are conceptually and technically narrowing the field: first, the community suffers from the lack of a common development platform to spread the latest advances, to consolidate prototypes, and to compare results among different research groups. Second, there has been an abundance of home-made solutions that inevitably led to a heterogeneity of technical approaches to the same problems and to an absence of standards, making the reuse of already developed and well-tested code problematic. Finally, recent research trends keep considering robotic devices as mere passive actuators of users' intentions by mostly neglecting the potential benefits of including robotic artificial intelligence in the decoding workflow. Furthermore, we speculate that the lack of technical tools (e.g., a common development ecosystem) might also conceptually affect the direction of the current neurorobotics research by slowing down the necessary integration between neural interfaces and robotics. It is worth mentioning that a variety of open-source platforms already exists in the neurorobotics field to acquire, process, and decode neurophysiological signals (e.g., LSL, BCI2000, OpenViBE, TOBI Common Implementation Platform, BioSig, BCILAB, BCI++, xBCI, BF++, PMW, and VETA Brunner et al., 2012; Stegman et al., 2020). Although each software has specific features and advantages, they only partially face all the aforementioned challenges. Furthermore, to the best of our knowledge, neither of them explicitly targets the integration of robotic platforms nor do they provide out-of-the-box solutions to directly interact with external devices.

In the current scenario, we firmly believe in the urgency of a common and open-source research framework for the future development of the neurorobotic field. Hence, we spotlight Robot Operating System (ROS)-Neuro, the first middleware explicitly devised to treat the multidisciplinary facets of neurorobotics with the same level of importance, to promote a holistic approach to the field, and to foster the research of a new generation of neural driven robotic devices.

2. ROS-NEURO MIDDLEWARE

2.1. Overview

ROS-Neuro has been designed to represent the first open-source neurorobotic middleware that places human neural interfaces and robotic systems at the same conceptual and implementation level. ROS-Neuro is an extension of ROS that for many years is considered the standard platform for robotics (Quigley et al., 2009). One of the strengths of ROS is its modularity and the possibility for different research groups to develop stand-alone components all relying on the same standard communication infrastructure. A similar requirement is a cornerstone for the workflow of any closed-loop neural interface where—for instance—acquisition, processing, and decoding methods should run in parallel in order to provide a continuous/discrete control signal to drive the robotic device. ROS-Neuro not only exploits such modular design but also provides several standard interfaces to acquire neurophysiological signals from different commercial devices to process EEG and EMG signals with traditional methods and to classify data with common machine learning algorithms. As in the case of ROS, the aim of ROS-Neuro is to allow the development of neurorobotic applications among different research groups as well as the possibility to easily compare heterogeneous methodological approaches and to rely and evaluate solutions proposed by others. This is guaranteed by its multi-process architecture where several stand-alone executables can coexist and can communicate through the provided network infrastructure. Moreover, each of these processes can be easily exchanged between research groups with the only requirement of sharing the same interface. The concept of ROS-Neuro has been introduced for the first time in Beraldo et al. (2018b) and in the following years, authors implemented and carefully tested packages to acquire, record, process, and visualize EEG and EMG data (Tonin et al., 2019; Beraldo et al., 2020). The aim of this contribution is to present ROS-Neuro to the community by providing a description of its main features and potentialities.

2.2. Abstraction, Modularity, and Parallel Architecture

Robotic applications and human neural interfaces share several similarities in the technical and implementation workflow. As robotics is traditionally based on the interactions between perception and planning and action, neural interfaces rely on the acquisition, processing, and classification closed-loop where the human plays the twofold role of generating the input signals and monitoring (as well as adapting to) the results of the decoding. Tonin and Millán (2021). ROS-Neuro generalizes such an architecture by providing modules to gather neurophysiological signals (`rosneuro_acquisition` package), to record the acquired data (`rosneuro_recorder`), to process and decode it (`rosneuro_buffers`, `rosneuro_filters`, `rosneuro_processing`), and to finally infer the intention of the user (`rosneuro_decisionmaking`). As in the case of the packages available in the ROS ecosystem, these modules represent generic interfaces that neither depends on specific hardware devices nor on particular processing methods. For instance, `rosneuro_acquisition` package is designed to work with plugins that can support different EEG/EMG devices and that can be independently developed (and shared) by any research group according to their needs. However, it is worth mentioning that ROS-Neuro already provides plugins that interface with the most used commercial acquisition systems (e.g., g.Tec, BioSemi, ANTNeuro, Cognionics). Similarly, packages like `rosneuro_buffers` and `rosneuro_filters` implement widely commonly used methods to process neural data such as spatial filters, DC removal algorithms, and windowing that can be easily extended and integrated with custom solutions provided by researchers. **Table 1** lists the acquisition systems (hardware devices and software platforms) compatible with

TABLE 1 | List of acquisition devices and platforms currently compatible with the `rosneuro_acquisition` package and the file formats supported by the `rosneuro_recorder`.

Hardware	Company	Driver	Plugin	Status
rosneuro_acquisition				
BioSemi ActiveTwo	BioSemi	free	rosneuro::EGDDevice	Tested
MindWave Headsets	Neurosky	free	rosneuro::EGDDevice	Untested
Bittium NeurOne	Bittium	free	rosneuro::EGDDevice	Untested
g.USBamp	g.Tec	proprietary	rosneuro::EGDDevice	Tested
g.NEEDaccess	g.Tex	proprietary	rosneuro::EGDDevice	Untested
BitBrain EEG	BitBrain	proprietary	rosneuro::EGDDevice	Untested
DSI-24	Wearable Sensing	proprietary	rosneuro::EGDDevice	Tested
CGX Quick-20	Cognionics	proprietary	rosneuro::EGDDevice	Tested
eego sport and mylab	AntNeuro	proprietary	rosneuro::EGDDevice	Tested
Ultracortex Mark IV	OpenBCI	free	rosneuro::LSLDevice	Tested
LabStreaming layer	/	free	rosneuro::LSLDevice	Tested
Tobi Interface A	/	free	rosneuro::EGDDevice	Untested
General data format (GDF) file	/	free	rosneuro::EGDDevice	Tested
BioSemi data format (BDF) file	/	free	rosneuro::EGDDevice	Tested
File format	**Company**	**Driver**		**Status**
rosneuro_recorder				
General data format (GDF)	/	free		Tested
BioSemi data format (BDF)	BioSemi	free		Tested
Filter	**Type**		**Class**	**Status**
rosneuro_filters				
DC removal	Temporal		rosneuro::Dc<T>	Tested
Common Average Reference	Spatial		rosneuro::Car<T>	Tested
Laplacian derivation	Spatial		rosneuro::Laplacian<T>	Tested
Blackman	Windowing		rosneuro::Blackman<T>	Tested
Flattop	Windowing		rosneuro::Flattop<T>	Tested
Hamming	Windowing		rosneuro::Hamming<T>	Tested
Hann	Windowing		rosneuro::Hann<T>	Tested
Buffer	**Type**		**Class**	**Status**
rosneuro_buffers				
RingBuffer	FIFO		rosneuro::RingBuffer<T>	Tested
Application	**Type**			**Status**
rosneuro_visualizer				
neuroviz	Temporal scope			Tested

The table also provides the filters and buffers available in the `rosneuro_filters` *and* `rosneuro_buffers` *packages. It is worth noticing that both filters and buffers can be easily concatenated via configuration file [please refer to* `FilterChain` *in Robot Operating System (ROS)]. Finally,* `neuroviz` *application is listed—the electroencephalography (EEG)/electromyography (EMG) scope provided by the* `rosneuro_visualizer` *package. In the last column, Untested status means that the related hardware is technically supported by the plugin but it was not possible to test it.*

ROS-Neuro and the supported file formats to store the acquired data. Furthermore, the filters, buffers, and the application scope provided by ROS-Neuro are reported.

Another feature of ROS-Neuro is the possibility to conveniently implement parallel pipelines with the minimum developing effort. This is of particular interest for many emerging aspects of hybrid neural interfaces. On the one hand, these interfaces are designed to simultaneously acquire, process, and fuse together heterogeneous neurophysiological signals from several sources [e.g., EEG, EMG, electrooculography (EOG)] in order to improve the robustness of the whole system (Müller-Putz et al., 2011). On the other, they can rely on different processing workflows to decode concurrent tasks performed by the user. In both cases, ROS-Neuro already exploits the ROS optimized communication infrastructure and it can rely on built-in solutions to synchronize and align data streams from different processes (e.g., hardware-based trigger) (Bilucaglia et al., 2020). This enormously facilitates the implementation of interfaces where—for instance—multiple acquisition processes are instantiated to simultaneously gather EEG and EMG signals (**Figure 1**). Then, specific processing may be applied to EEG in order to decode the intention of the user to reach an object with a robotic arm or an upper-limb exoskeleton; at the same time, residual muscular activity may be exploited to distinguish the type of grasping. Furthermore, brain signals can be also analyzed in conjunction with environmental information in order to recognize potential erroneous actions performed by the neuroprosthesis.

2.3. Standard Messages and Communication

The rapid growth of the neurorobotics field, and in particular, of human neural interfaces has led to the heterogeneity of technical solutions. In this scenario, one of the main limitations of current developing frameworks is the custom approaches to sharing information between the different modules composing the closed-loop implementation of neural interfaces. Traditionally, each research group relies on its own data structures to represent neurophysiological data and custom-made network infrastructures to implement the communication between the several processing steps. Such a lack of a common approach strongly downplays the impact of the technology by limiting the possibility to share developing tools, to exploit solutions already implemented, and to replicate results achieved by different research groups.

ROS-Neuro provides standard messages to exchange data structures between the modules usually implemented within neurorobotics applications. Moreover, messages are available to all modules by the ROS network infrastructure-based peer-to-peer communication mechanisms. Data acquired by the `rosneuro_acquisition` is streamed as `NeuroFrame` messages within the ecosystem (**Figure 1**), where several modules can subscribe to the stream at the same time and concurrently process the messages in order to extract and decode heterogeneous features from neurophysiological signals. Similarly, the output of the decoder is translated into `NeuroPrediction` messages that can be exploited to directly control the robotic application or to be further processed. Furthermore, it is worth mentioning that ROS allows to quickly extend the interface of any message without the need for coding in order to handle specific, application-related requirements. As a consequence, ROS-Neuro not only offers the possibility to conveniently compare different methodological approaches even during closed-loop operations but also to effortlessly distribute implementation solutions among different research groups with the only requirement of providing the standard message interface.

2.4. Robotic Devices

The straightforward integration between neural interfaces and external actuators is the most evident advantage of ROS-Neuro middleware. Traditionally, the inclusion of robotic devices has been considered a pure technical challenge, and thus, a variety of home-made solutions has been adopted to deliver the output of the neural interface to the robot ecosystem. However, the drawback of this approach is twofold: first, from an engineering perspective, custom solutions are often not optimized and efficient with the consequence of an increased risk of technical faults. Second, the communication stream between neural interfaces and robotic devices is usually limited to a single uni-dimensional control signal. This definitely narrows the research on new human-machine interaction (HMI) modalities and the introduction of bidirectional communication with the robot to enhance the robustness and the reliability of the whole system. For instance, a robot's intelligence may provide information about the operational context to the neural interface in order to modulate the velocity of the decoder response, thus facilitating the control or preventing the delivery of an erroneous command according to the current situation. Thus, the level of autonomy of the neurorobotic device may be changed in the case, for example, a smart wheelchair crosses a narrow passage or a robotic hand attempts to grasp an unusual-shaped object (**Figure 1**).

By construction, ROS-Neuro explicitly provides such a common and bidirectional communication between the neural interface workflow and the robotic intelligence by exploiting the ROS ecosystem and the several packages already available in the ROS community. Furthermore, the reliability and robustness of the communication is guaranteed by the ROS network infrastructure by reducing the likelihood of technical shortcomings and malfunctions.

3. EVALUATION OF ROS-NEURO: THE CYBATHLON EVENT

ROS-Neuro has been evaluated by using different hardware devices (e.g., a variety of commercial EEG/EMG amplifiers and various robotic platforms Beraldo et al., 2018a,b, 2020; Tonin et al., 2019) during several experiments. In all cases, ROS-Neuro demonstrated its flexibility, reliability, and robustness. However, the most critical stress test for ROS-Neuro has been the usage for the Cybathlon events (Wolf and Riener, 2018). Cybathlon is the first neurorobotic championship where several international teams from all over the world competed in different disciplines: from races with lower and upper limb prostheses to races with wheelchairs and exoskeletons. The ultimate goal of Cybathlon is to foster the research and development of daily-life solutions for people with disabilities. In this context, one of the most challenging disciplines was the BCI Race[1] where pilots with a severe motor disability (i.e., inclusion criteria ASIA-C) exploited a non-invasive BMI to control an avatar on the screen during a virtual race. Authors participated in the Cybathlon BCI Series 2019 and the Cybathlon 2020 Global Edition with the WHi Team composed of researchers from the University of Padua (Italy). In these periods and in the related longitudinal training of the pilot, ROS-Neuro has been extensively used and tested. In both

[1]https://cybathlon.ethz.ch/en/event/disciplines/bci

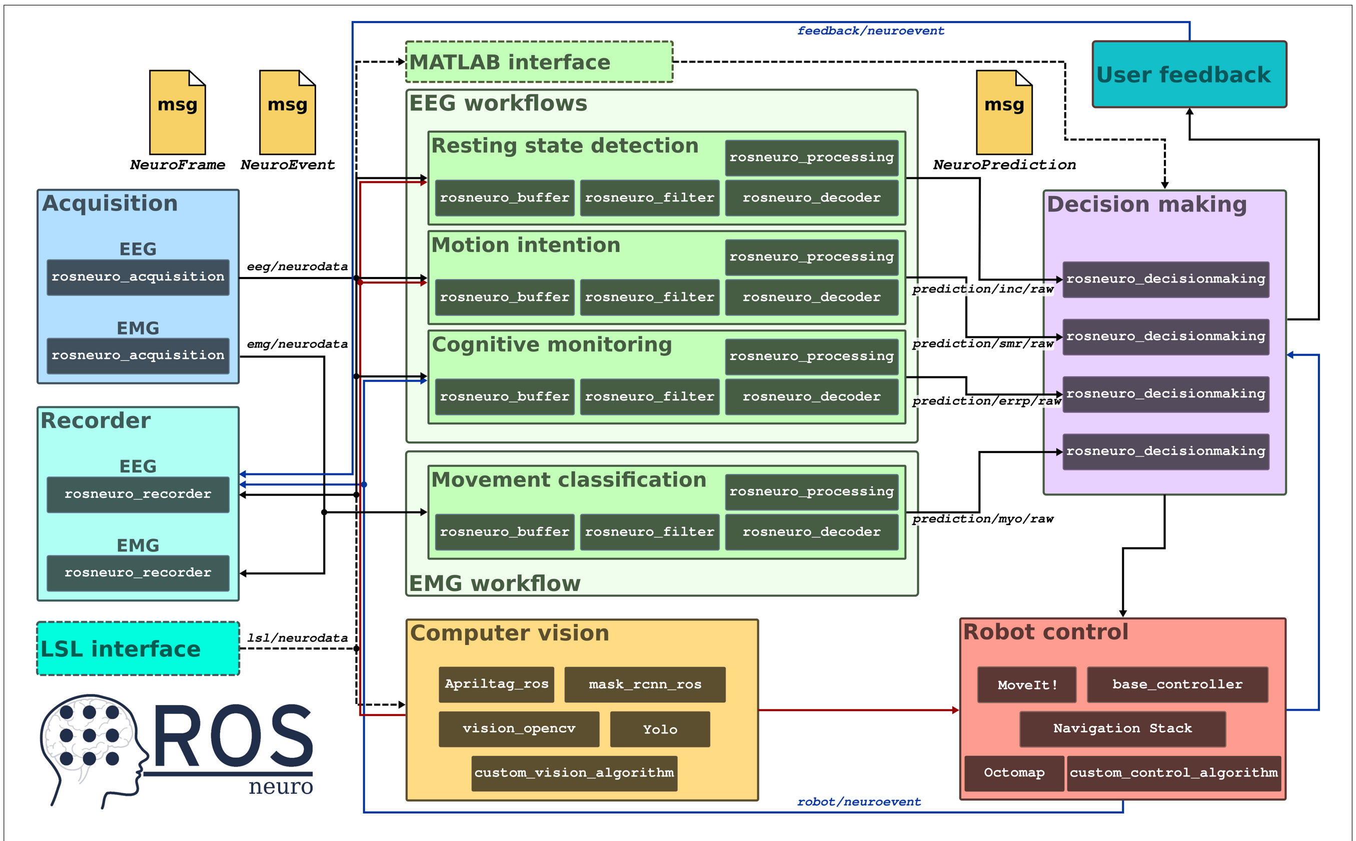


FIGURE 1 | A schematic representation of a hybrid, multi-process implementation of a neural interface with Robot Operating System (ROS)-Neuro. Two acquisition systems are used in parallel to acquire (and record) EEG and EMG data (blue and cyan boxes). An additional interface can be added to record the data stream from the LSL device (dashed cyan box). Data is made available in the `eeg/neurodata` and `emg/neurodata` communication channels as `NeuroFrame` messages to all the other modules. In the example, four different workflows work in parallel (green boxes) to detect resting state, motion intention, to monitor the behavior of the system from EEG signals, and to classify residual muscular activity from EMG data. An additional processing module can be added by exploiting the ROS-Neuro MATLAB interface (dashed green box). The output of the processing workflows is published as `NeuroPrediction` messages in the `prediction/*/raw`. A decision making module (purple box) reads the predicted outputs and generates a proper control signal for the robotic device. Such a signal can be also used to provide feedback to the user. In parallel, computer vision algorithms and ROS navigation packages (red and yellow boxes) not only take care of controlling the robot but also provide environmental information for the EEG workflows (red and blue arrows).

editions, the WHi Team won the gold medal by awarding the race records. Most importantly, ROS-Neuro was confirmed to be reliable and robust during the whole training and, especially, in the demanding conditions of the event. Neither technical faults nor difficulties or glitches during the interface with the official Cybathlon infrastructure (for connecting to the virtual race) have been reported. We speculate that the efficiency, the flexibility, and the performance of ROS-Neuro were one of the key reasons (among others) for the success of the WHi Team at the Cybathlon.

4. DISCUSSION

Recent evidence in literature highlighted the importance of reconsidering the current approach to neurorobotics in order to enhance the reliability of neural driven robotic devices, and thus, foster the translational impact and the daily usage of the technology (Perdikis et al., 2018; Perdikis and Millán, 2020; Tonin and Millán, 2021). In particular, the research community started following a more holistic approach by investigating the mutual interactions between the actors of the system, i.e., the user, the decoder, and the robotic device. For instance, several studies have demonstrated the key role of mutual learning between user and decoder to facilitate the acquisition of BMI skills and enhance the reliability of BMI-driven devices (Perdikis and Millán, 2020). Similarly, it has been shown that a neural interface explicitly designed to promote the interaction between user and robotic intelligence can support a more natural and efficient control of the device (Tonin et al., 2020). In this scenario, we speculate that ROS-Neuro might offer the technical counterpart of this new research direction by not only allowing to develop the neural interface workflow and the robotic intelligence within the same ecosystem but also by guaranteeing high performance and strong robustness of the whole application.

Although we previously pinpointed ROS-Neuro features with respect to the current platforms available in the community, it is worth mentioning that it should not be considered a direct competitor. Indeed, ROS-Neuro represents uniqueness in the neurorobotics field with the explicit aim of integrating neural interfaces and robotics by exploiting the advantages of both fields. Furthermore, current development frameworks to acquire neural signals can easily be included in the ROS-Neuro infrastructure, for instance, plugin to connect LSL is already implemented and available in the public repository to incorporate external information streams into the ROS-Neuro ecosystem.

ROS-Neuro is distributed as an open-source project, and it is available on GitHub[2]. As in the case of ROS, the success of ROS-Neuro strictly depends on the creation of a wide community disseminating the latest developments and including the multidisciplinary needs of the different research groups. It is our opinion that ROS-Neuro represents the only way to achieve a robust and flexible ecosystem, to review and evaluate alternative approaches, and, finally, to boost neurorobotics technology. ROS-Neuro supports the development of packages in C++ and Python, and we acknowledge that this might hinder the approach to the platform, especially if researchers are used to working with GUI-based software (e.g., OpenVibe). For this reason, ROS-Neuro already provides a MATLAB interface (`rosneuro_matlab`) in order to facilitate the integration with toolboxes widely spread in the community and to mitigate the effort of those people not used to such programming languages. Nevertheless, we consider that this is a small price to pay in comparison with the advantages in terms of reliability, performance, and integration that ROS-Neuro can offer.

Finally, the current version of ROS-Neuro is fully based on ROS 1 LTS (ROS Noetic Ninjemys)[3], and thus, it works on Ubuntu Linux operating systems only. However, in a few years, the community started the development of ROS 2 that—among several changes—is the first multi-platform version of ROS (i.e., on Ubuntu Linux, MacOS, and Windows 10). The transition of ROS-Neuro from ROS 1 to ROS 2 has already been scheduled to expand the base of potential users of ROS-Neuro. Nevertheless, the effort to develop and maintain both versions can be demanding, and it would be beneficial to have the support of the whole community.

In conclusion, we firmly believe that ROS-Neuro might be the future development platform for neurorobotics. Furthermore, as in the case of ROS, it might represent the starting point for the creation of a flourishing research community to foster the translational impact of neurorobotics technology.

[2]https://github.com/rosneuro
[3]https://www.ros.org/

DATA AVAILABILITY STATEMENT

Publicly available code was reported in this study. This code can be found here: GitHub, https://github.com/rosneuro.

AUTHOR CONTRIBUTIONS

LT and EM conceived the idea of ROS-Neuro. All authors wrote, reviewed, and approved the final manuscript.

FUNDING

Part of this work was supported by MUR (Italian Minister for University and Research), under the initiative Departments of Excellence (Law 232/2016), and by the Department of Information Engineering, University of Padova, under the BrainGear project (TONI_BIRD2020_01).

ACKNOWLEDGMENTS

We would like to thank Vassilli srl for sponsorship and hardware support to the WHi team participating in the Cybathlon competition.

REFERENCES

Aflalo, T., Kellis, S., Klaes, C., Lee, B., Shi, Y., Pejsa, K., et al. (2015). Decoding motor imagery from the posterior parietal cortex of a tetraplegic human. *Science* 348, 906–910. doi: 10.1126/science.aaa5417

Batzianoulis, I., El-Khoury, S., Pirondini, E., Coscia, M., Micera, S., and Billard, A. (2017). Emg-based decoding of grasp gestures in reaching-to-grasping motions. *Rob. Auton. Syst.* 91, 59–70. doi: 10.1016/j.robot.2016.12.014

Batzianoulis, I., Krausz, N. E., Simon, A. M., Hargrove, L., and Billard, A. (2018). Decoding the grasping intention from electromyography during reaching motions. *J. Neuroeng. Rehabil.* 15, 57. doi: 10.1186/s12984-018-0396-5

Beraldo, G., Antonello, M., Cimolato, A., Menegatti, E., and Tonin, L. (2018a). "Brain-computer interface meets ROS: a robotic approach to mentally drive telepresence robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, QLD: IEEE), 4459–4464.

Beraldo, G., Castaman, N., Bortoletto, R., Pagello, E., Milln, J., d,. R., et al. (2018b). "ROS-health: an open-source framework for neurorobotics," in *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)* (Brisbane, QLD: IEEE), 174–179.

Beraldo, G., Tortora, S., Menegatti, E., and Tonin, L. (2020). "ROS-Neuro: implementation of a closed-loop BMI based on motor imagery," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (Toronto, ON: IEEE), 2031–2037.

Betti, S., Zani, G., Guerra, S., Castiello, U., and Sartori, L. (2018). Reach-to-grasp movements: a multimodal techniques study. *Front. Psychol.* 9, 990. doi: 10.3389/fpsyg.2018.00990

Bilucaglia, M., Masi, R., Stanislao, G. D., Laureanti, R., Fici, A., Circi, R., et al. (2020). ESB: a low-cost EEG synchronization box. *HardwareX* 8:e00125. doi: 10.1016/j.ohx.2020.e00125

Borton, D., Micera, S., Millán, J., and Courtine, G. (2013). Personalized neuroprosthetics. *Sci. Transl. Med.* 5, 210rv2. doi: 10.1126/scitranslmed.3005968

Brunner, C., Andreoni, G., Bianchi, L., Blankertz, B., Breitwieser, C., Kanoh, S., et al. (2012). "BCI software platforms," in *Towards Practical Brain-Computer Interfaces. Biological and Medical Physics, Biomedical Engineering*, eds B. Allison, S. Dunne, R. Leeb, R. Del, J. Millán, and A. Nijholt (Berlin; Heidelberg: Springer). doi: 10.1007/978-3-642-29746-5_16

Castellini, C., Gruppioni, E., Davalli, A., and Sandini, G. (2009). Fine detection of grasp force and posture by amputees via surface electromyography. *J. Physiol. Paris* 103, 255–262. doi: 10.1016/j.jphysparis.2009.08.008

Chaudhary, U., Birbaumer, N., and Ramos-Murguialday, A. (2016). Brain-computer interfaces for communication and rehabilitation. *Nat. Rev. Neurol.* 12, 513–525. doi: 10.1038/nrneurol.2016.113

Cipriani, C., Antfolk, C., Controzzi, M., Lundborg, G., Rosen, B., Carrozza, M. C., et al. (2011). Online myoelectric control of a dexterous hand prosthesis by transradial amputees. *IEEE Trans. Neural Syst. Rehabil. Eng.* 19, 260–270. doi: 10.1109/TNSRE.2011.2108667

De Luca, A., Bellitto, A., Mandraccia, S., Marchesi, G., Pellegrino, L., Coscia, M., et al. (2019). Exoskeleton for gait rehabilitation: effects of assistance, mechanical structure, and walking aids on muscle activations. *Appl. Sci.* 9, 2868. doi: 10.3390/app9142868

Edelman, B. J., Meng, J., Suma, D., Zurn, C., Nagarajan, E., Baxter, B. S., et al. (2019). Noninvasive neuroimaging enhances continuous neural tracking for robotic device control. *Sci. Rob.* 4, eaaw6844. doi: 10.1126/scirobotics.aaw6844

Farrell, T. R., and Weir, R. F. (2008). A comparison of the effects of electrode implantation and targeting on pattern classification accuracy for prosthesis control. *IEEE Trans. Biomed. Eng.* 55, 2198–2211. doi: 10.1109/TBME.2008.923917

Hochberg, L., Bacher, D., Jarosiewicz, B., Masse, N., Simeral, J., Vogel, J., et al. (2012). Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature* 485, 372–375. doi: 10.1038/nature11076

Iez, E., Azor-n, J. M., beda, A., Ferrndez, J. M., and Fernndez, E. (2010). Mental tasks-based brain robot interface. *Rob. Auton. Syst.* 58, 1238–1245. doi: 10.1016/j.robot.2010.08.007

Kennedy, P., and Bakay, R. (1998). Restoration of neural output from a paralyzed patient by a direct brain connection. *Neuroreport* 9, 1707–1711. doi: 10.1097/00001756-199806010-00007

Leeb, R., Perdikis, S., Tonin, L., Biasiucci, A., Tavella, M., Creatura, M., et al. (2013). Transferring brain-computer interfaces beyond the laboratory: successful application control for motor-disabled users. *Artif. Intell. Med.* 59, 121–132. doi: 10.1016/j.artmed.2013.08.004

Leeb, R., Tonin, L., Rohm, M., Desideri, L., Carlson, T., and Millán, J. (2015). Towards independence: a BCI telepresence robot for people with severe motor disabilities. *Proc. IEEE* 103, 969–982. doi: 10.1109/JPROC.2015.2419736

Liu, D., Chen, W., Lee, K., Chavarriaga, R., Bouri, M., Pei, Z., et al. (2017). Brain-actuated gait trainer with visual and proprioceptive feedback. *J. Neural Eng.* 14, 056017. doi: 10.1088/1741-2552/aa7df9

Liu, D., Chen, W., Lee, K., Chavarriaga, R., Iwane, F., Bouri, M., et al. (2018). EEG-based lower-limb movement onset decoding: continuous classification and asynchronous detection. *IEEE Trans. Neural Syst. Rehabil. Eng.* 26, 1626–1635. doi: 10.1109/TNSRE.2018.2855053

Müller-Putz, G., Breitwieser, C., Cincotti, F., Leeb, R., Schreuder, M., Leotta, F., et al. (2011). Tools for Brain-Computer Interaction: a general concept for a hybrid bci. *Front. Neuroinform.* 5, 30. doi: 10.3389/fninf.2011.00030

Parajuli, N., Sreenivasan, N., Bifulco, P., Cesarelli, M., Savino, S., Niola, V., et al. (2019). Real-time EMG based pattern recognition control for hand prostheses: a review on existing methods, challenges and future implementation. *Sensors* 19, 4596. doi: 10.3390/s19204596

Perdikis, S., and Millán, J. (2020). Brain-machine interfaces: a tale of two learners. *IEEE Syst. Man Cybern. Mag.* 6, 12–19. doi: 10.1109/MSMC.2019.2958200

Perdikis, S., Tonin, L., Saeedi, S., Schneider, C., and Millán, J. (2018). The cybathlon BCI race: successful longitudinal mutual learning with two tetraplegic users. *PLoS Biol.* 16, e2003787. doi: 10.1371/journal.pbio.2003787

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., et al. (2009). "ROS: an open-source Robot Operating System," in *ICRA workshop on Open Source Software, Vol. 3* (Kobe).

Stegman, P., Crawford, C. S., Andujar, M., Nijholt, A., and Gilbert, J. E. (2020). Brain computer interface software: a review and discussion. *IEEE Trans. Hum. Mach. Syst.* 50, 101–115. doi: 10.1109/THMS.2020.2968411

Sylos-Labini, F., La Scaleia, V., d'Avella, A., Pisotta, I., Tamburella, F., Scivoletto, G., et al. (2014). EMG patterns during assisted walking in the exoskeleton. *Front. Hum. Neurosci.* 8, 423. doi: 10.3389/fnhum.2014.00423

Tonin, L., Bauer, F., and Millán, J. (2020). The role of the control framework for continuous tele-operation of a BMI driven mobile robot. *IEEE Trans. Rob.* 36, 78–91. doi: 10.1109/TRO.2019.2943072

Tonin, L., Beraldo, G., Tortora, S., Tagliapietra, L., Milln, J., d,. R., et al. (2019). ROS-"Neuro: a common middleware for BMI and robotics. the acquisition and recorder packages," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (Bari: IEEE), 2767–2772.

Tonin, L., and Millán, J. (2021). Noninvasive brain-machine interfaces for robotic devices. *Ann. Rev. Control Rob. Auton. Syst.* 4, 191–214. doi: 10.1146/annurev-control-012720-093904

Wolf, P., and Riener, R. (2018). Cybathlon: how to promote the development of assistive technologies. *Sci. Rob.* 3, eaat7174. doi: 10.1126/scirobotics.aat7174

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

frontiers | Frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 23 May 2022
doi: 10.3389/fnbot.2022.845955

Continual Sequence Modeling With Predictive Coding

Louis Annabi[1]*, *Alexandre Pitti*[1] and *Mathias Quoy*[1,2]

[1] UMR8051 Equipes Traitement de l'Information et Systemes (ETIS), CY University, ENSEA, CNRS, Cergy-Pontoise, France,
[2] IPAL CNRS Singapore, Singapore, Singapore

Recurrent neural networks (RNNs) have been proved very successful at modeling sequential data such as language or motions. However, these successes rely on the use of the backpropagation through time (BPTT) algorithm, batch training, and the hypothesis that all the training data are available at the same time. In contrast, the field of developmental robotics aims at uncovering lifelong learning mechanisms that could allow embodied machines to learn and stabilize knowledge in continuously evolving environments. In this article, we investigate different RNN designs and learning methods, that we evaluate in a continual learning setting. The generative modeling task consists in learning to generate 20 continuous trajectories that are presented sequentially to the learning algorithms. Each method is evaluated according to the average prediction error over the 20 trajectories obtained after complete training. This study focuses on learning algorithms with low memory requirements, that do not need to store past information to update their parameters. Our experiments identify two approaches especially fit for this task: conceptors and predictive coding. We suggest combining these two mechanisms into a new proposed model that we label PC-Conceptors that outperforms the other methods presented in this study.

Keywords: predictive coding, continual learning, Reservoir Computing (RC), recurrent neural networks (RNN), conceptors

OPEN ACCESS

Edited by:
Jun Tani,
Okinawa Institute of Science and Technology Graduate University, Japan

Reviewed by:
Andrea Cossu,
University of Pisa, Italy
Yulia Sandamirskaya,
Intel, Germany

****Correspondence:***
Louis Annabi
louis.annabi@gmail.com

Received: *30 December 2021*
Accepted: *19 April 2022*
Published: *23 May 2022*

Citation:
Annabi L, Pitti A and Quoy M (2022) Continual Sequence Modeling With Predictive Coding. Front. Neurorobot. 16:845955. doi: 10.3389/fnbot.2022.845955

1. INTRODUCTION

Continual learning is a branch of machine learning aiming at equipping learning agents with the ability to learn incrementally without forgetting previously acquired knowledge. The continual learning setting typically involves several separate tasks where we assume data to be independent and identically distributed. The learning algorithm is confronted with each source of data (i.e., each task) sequentially. After a set amount of time on a task, training switches to a new task. This process is repeated until the learning algorithm has been confronted with all tasks.

Learning methods based on iterative updates of model parameters, such as the backpropagation algorithm, can be performed sequentially as new data becomes available. However, these methods might suffer from the problem known as catastrophic forgetting (McCloskey and Cohen, 1989) if the distribution of the data they process evolves over time. When adapting to the new task, they automatically overwrite the model parameters that were optimized according to the previous tasks. This is an important issue since it prevents artificial neural networks from being trained incrementally.

In this study, we focus on the problem of learning a repertoire of trajectories. As such, the training examples in each task are sequences that the learning algorithm has to generate from a discrete input (i.e., the index of the sequence). We study different Recurrent Neural Network (RNN) designs and learning algorithms for this continual learning task. We limit our comparison to models with low memory requirements and, thus, impose that at each time step t, the neural network computations and learning can only access the currently available quantities. In our case, these quantities are the currently hidden variables of the models, and the target output $\boldsymbol{x}_t^*$ provided by the data set. Consequently, learning methods based on BPTT do not qualify for this criterion, as they need to store in memory the past activations of the RNN hidden states to compute gradients. The advantage of models fitting this criterion is that they could in principle be implemented on dedicated hardware reproducing the neural network architecture, with no need for an external memory storing past inputs and activations.

To avoid confusion about the use of the word "online," we rather talk about *continual* learning to refer to the task temporality, and talk about *online* learning to refer to the sequence (the training example) temporality. The models studied in this section are thus trained both in a continual learning setting, since the different target trajectories are provided sequentially to the agent, and using online learning mechanisms since the algorithms for learning do not rely on a memory of past activations. The article is structured as follows: in Section 2, we review methods that have been proposed to mitigate the problem of catastrophic forgetting in artificial neural networks, as well as learning algorithms that can be performed online. In Section 3, we describe the experimental setting and the different algorithms, and present the obtained results in Section 4. Finally, in Section 5, we discuss our results in order to identify the online learning mechanisms for RNNs most suited for the continual learning of a repertoire of sequences.

2. RELATED WORK

There exists a large spectrum of methods to mitigate catastrophic forgetting in continual learning settings. Regularization methods typically aim at limiting forgetting by constraining learning with, e.g., sparsity constraints, early stopping, or identified synaptic weights that should not be overwritten. For instance, in Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), the update rule contains a regularization term that pulls the synaptic weights toward the optimal weights found for previous tasks, with a strength depending on the estimated importance of each synaptic weight.

Another approach is to rely on architecture modifications when new tasks are presented, for instance by freezing some of the previously learned weights (Mallya et al., 2018), or by adding new neurons and synaptic connections to the model (Li and Hoiem, 2017). Finally, rehearsal (Rebuffi et al., 2017) and generative replay (Shin et al., 2017) methods rely on saving examples or modeling past tasks for future use. By inserting training examples from the previous tasks, either saved or replayed, into the current task, these methods allow to retrain on those data points and thus limiting catastrophic forgetting.

In this study, we compare learning algorithms with low memory requirements in a continual learning setting. As such, we disregard approaches such as rehearsal and generative replay and only consider some simple regularization or architectural techniques to improve the performance of sequence memory models in a continual learning setting.

Many alternatives to BPTT have been investigated in the past decades, often with the goal of avoiding the problems known as exploding and vanishing gradients that can arise when using this learning algorithm (Pascanu et al., 2013). Here, we study two alternative approaches, namely, learning with evolution strategies, and Reservoir Computing (RC) (Verstraeten et al., 2007; Lukosevicius and Jaeger, 2009).

Using evolution strategies allows for learning RNN parameters without having to rely on past activations. The success of a certain parameter configuration can be measured online, for instance by comparing the network's output at each time step t with the target output. Then, this score is used as the fitness measure to be minimized by evolution. Following this approach, Schmidhuber et al. (2005) and Schmidhuber et al. (2007) co-evolve different groups of neurons in a Long Short-Term Memory (LSTM) network. A similar approach is used by Pitti et al. (2017), where the fitness measure is used to directly optimize the inputs of an RNN.

Completely avoiding the problem of learning recurrent weights, a family of approaches has emerged in parallel with the field of computational neurosciences in the form of Liquid State Machines (Maass et al., 2002), and from the field of machine learning in the form of Echo State Networks (ESN) (Jaeger, 2001). These models, later brought together under the label of Reservoir Computing (Verstraeten et al., 2007; Lukosevicius and Jaeger, 2009), discards the difficulties of learning recurrent weights by instead developing techniques to find relevant initializations of these parameters.

Typically, the recurrent connections are set in order for the RNN to exhibit rich non-linear (and sometimes self-sustained) dynamics, that are decoded by a learned readout layer. If the dynamics of the RNN activation are complex enough (e.g., they do not converge too rapidly toward a point attractor or limit cycle attractor), various output sequences can be decoded from those. Training RC models then come down to learning the weights of the readout layer, which is an easier optimization problem that can be tackled with several algorithms. This output layer can, e.g., be trained using stochastic gradient descent, without the need for BP. The FORCE algorithm (Sussillo and Abbott, 2009) improves this learning by running an iterative estimate of the correlation matrix of the hidden state activations.

Another interesting learning mechanism is presented in Jaeger (2014a,b) under the name of Conceptors. This method exploits the fact that the hidden state dynamics triggered by an input pattern is typically bounded to a certain subspace of lower dimension. By identifying the subspace for each possible input pattern, it is possible to decorrelate the training of each target trajectory by focusing learning on the readout connections that come from the corresponding hidden state subspace (called

Conceptor). This method allows training a sequence memory where the learning of a new pattern has limited interference with already learned ones.

Finally, the Predictive Coding (PC) theory (Rao and Ballard, 1999; Clark, 2013) also provides learning rules that do not rely on past activations. According to PC, prediction error neurons are intertwined with the neural generative model and encode at each layer the discrepancy between the top-down prediction and the current representation. It has been shown that this construction allows propagating the output prediction error information back into the generative model and even approximates the backpropagation algorithm (Whittington and Bogacz, 2017; Millidge et al., 2020).

Taking inspiration from the PC theory, we propose several models that integrate prediction error neurons into a simple RNN design. These prediction error neurons transport the error information from the output layer to the hidden layer, which provides a local target that can be used to learn the recurrent and input weights. We label the resulting models PC-RNN (for Predictive Coding Recurrent Neural Network). In Appendix A, we show how these models can be derived from the application of gradient descent on a quantity called variational free-energy expressed according to different generative models. The resulting models slightly deviate from other approaches such as the Parallel Temporal Coding Network (P-TNCN) described in Ororbia et al. (2020), and the original PC model presented in Rao and Ballard (1997), which suggests learning feedback weights responsibly for the bottom-up computations instead of copying the forward weights, as performed in the proposed models.

There have been other evaluations of continual learning methods applied to RNNs (Sodhani et al., 2020; Cossu et al., 2021b), some even focusing on ESNs (Cossu et al., 2021a). While these studies compare many continual learning techniques, they do not consider the online learning constraint, and almost exclusively focus on sequence classification tasks. In contrast, this study investigates continual learning methods that can be used online, applied to the incremental learning of a repertoire of trajectories.

3. MATERIALS AND METHODS

In this section, we detail our experimental setting as well as the different models that we use for the comparative study.

3.1. Experimental Setting

Each RNN model is trained sequentially on p sequence generation tasks. The p sequences to be learned are sampled from a data set of motion capture trajectories of dimension 62. Each point $\boldsymbol{x}_t^*$ describes a body configuration, as represented in **Figure 1**. These trajectories were obtained from the CMU Motion Capture Database. We make a distinction between the validation set used to optimize the hyperparameters of each model, and the test set, used to measure the performance of each model. In our experiments, the validation set is composed of $p = 15$ trajectories of a subject (#86 in the database) practicing various sports. The test set is composed of $p = 20$ trajectories of a subject

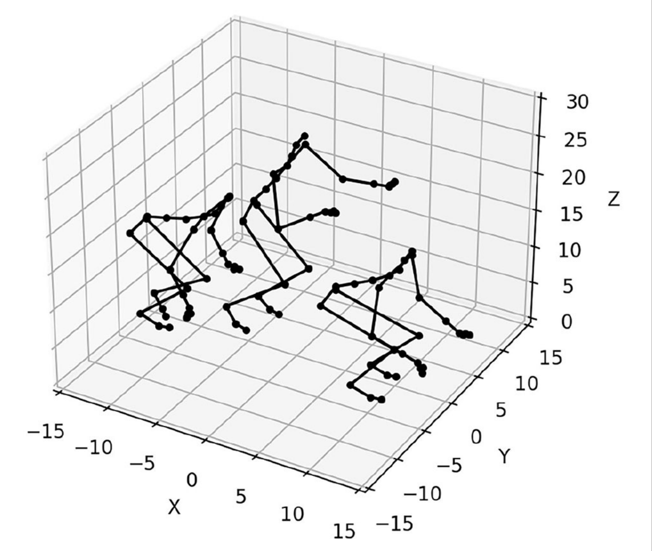


FIGURE 1 | Three body configurations taken from a trajectory capturing a jump motion.

(#62 in the database) performing construction work (screwing, hammering, etc.).

We also measure the performance of each model on a different test set of $p = 20$ simple 2D trajectories corresponding to handwritten letters taken from the UCI Machine Learning Repository (Dua and Graff, 2019). All trajectories are resampled to last for 60 time steps. These data sets were chosen since they represent potential use cases of the models presented in this work. For instance, the proposed continual learning algorithms could be used to incrementally train a robot manipulator to perform certain motor trajectories.

We assume that the model knows when a transition between two tasks occurs, and provide to the RNN the current task index k as a one-hot vector input of dimension p. Otherwise, this distributional shift could, e.g., be automatically detected through a significant increase of the prediction error.

The end goal of this experiment is to identify online learning mechanisms for RNNs that extend properly to the continual learning case. The RNN architectures typically comprise three types of weight parameters to be learned: the output weights, the recurrent weights, and the input weights. As such, we split our analysis into three comparisons focusing on the learning methods for each type of parameter.

For each learning mechanism, we perform an optimization of hyperparameters using Bayesian optimization with Gaussian processes and Matern 5/2 Kernel, similarly to the RNN encoding capacity comparative analysis performed in Collins et al. (2016).

This method tries to approximate the function $\boldsymbol{P} \rightarrow f(\boldsymbol{P})$ that associates a scoring function with a certain hyperparameter configuration $\boldsymbol{P}$. This approximation is estimated based on points $\big((\boldsymbol{P}_0, f(\boldsymbol{P}_0)), (\boldsymbol{P}_1, f(\boldsymbol{P}_1)), \cdots\big)$ sampled sequentially by the optimizer. The function used by the optimizer to guide its sampling process is called acquisition function. Here, we used an expected improvement acquisition function, meaning that at each iteration, the optimizer samples the point P which is most likely to improve the current estimated maximum of the function f. Compared to exhaustive hyperparameter optimization methods such as random search or grid search, this method is expected to converge faster and to better configurations. To perform this hyperparameter optimization we used the `gp_minimize` function from the scikit-optimize library in python.

The hyperparameters of the models are optimized in order to minimize the final average prediction error on the p target sequences of the validation set. For each model, the hyperparameters to optimize are the learning rates associated with the input, recurrent, and output weights, as well as some other coefficients specific to certain learning algorithms. The score function associates each hyperparameter configuration with a real-valued score computed as the negative logarithm of the average prediction error at the end of training.

With the hyperparameter configurations we obtain, we perform for each model 10 seeds of training in the continual learning setting to measure their performances. The final average prediction error on the p sequences can be used to evaluate and compare the different learning mechanisms.

3.2. Benchmark Models

The models for this benchmark were chosen in order to identify the relevant mechanisms for training RNNs in a continual learning setting. As already said, we also limit this analysis to learning algorithms that can be performed *online*, i.e., without relying on past activations. For each set of weights, we compare the different models listed in **Table 1**.

3.2.1. Output Weights

For the learning of the output weights of RNNs, denoted $\boldsymbol{W}_o$, we compared four learning methods, applied to the simple RNN architecture represented in **Figure 2**. All methods share the same architecture, and do not provide any learning mechanism for the recurrent weights. At each time step, the hidden state and output prediction are obtained with the following equations:

$$\boldsymbol{h}_t = (1 - \frac{1}{\tau})\boldsymbol{h}_{t-1} + \frac{1}{\tau}\boldsymbol{W}_r \cdot \tanh(\boldsymbol{h}_{t-1}) \quad (1)$$

$$\boldsymbol{x}_t = \boldsymbol{W}_o \cdot \tanh(\boldsymbol{h}_t) \quad (2)$$

where τ is a time constant controlling the velocity of the hidden state dynamics.

The four methods differ with regard to the learning mechanism applied to the output weights. First, output weights can be learned using standard stochastic gradient descent. In the RNN models we consider, the prediction $\boldsymbol{x}_t$ is not re-injected into the recurrent computations. As such, the output weights gradients can be computed using only the target signal $\boldsymbol{x}_t^*$, the prediction $\boldsymbol{x}_t$, and the hidden state $\boldsymbol{h}_t$. These computations do not involve the backpropagation of a gradient through time and thus qualify as an online learning method. This first learning rule, also known as the Widrow-Hoff learning rule is expressed as:

$$\boldsymbol{W}_o \leftarrow \boldsymbol{W}_o + \lambda \boldsymbol{\epsilon}_{x,t} \cdot \tanh(\boldsymbol{h}_t)^{\mathsf{T}} \quad (3)$$

where λ is the learning rate of the output weights, and $\boldsymbol{\epsilon}_{x,t}$ is the prediction error on the output layer, i.e., the difference $\boldsymbol{x}_t^* - \boldsymbol{x}_t$.

The second learning mechanism that we study is stochastic gradient descent aided by Conceptors (Jaeger, 2014a,b). Mathematically, this method can be implemented using only online computations. The Conceptor $\boldsymbol{C}$ associated with some input can be defined as the matrix corresponding to a soft projection on the subspace where the hidden state dynamics lie when stimulated with this input. The softness of this projection is controlled by a positive parameter α called the aperture. This matrix $\boldsymbol{C}$ can be computed using the hidden state correlation matrix $\boldsymbol{R}$ estimated online based on the hidden state dynamics:

$$\boldsymbol{C} = \boldsymbol{R} \cdot (\boldsymbol{R} + \alpha^{-2}\mathbb{I})^{-1} \quad (4)$$

$$\boldsymbol{R}_{t+1} = \big(1 - \frac{1}{t+1}\big)\boldsymbol{R}_t + \frac{1}{t+1}\big(\boldsymbol{h}_t \cdot \boldsymbol{h}_t^{\mathsf{T}}\big) \quad (5)$$

In a continual learning setting, for each new task, we can compute the Conceptor corresponding to the complement of the subspace where the previously seen hidden states lie, as $\mathbb{I} - \boldsymbol{C}$. This Conceptor is used to project the new hidden states into a subspace orthogonal to the subspace in which lie the previously seen

TABLE 1 | Summary of the models used in our benchmark.

Weights	Model
Output weights	ESN (Widrow-Hoff)
	Conceptors
	EWC
	ESN + GR
Recurrent weights	PC-RNN-V
	P-TNCN
	PC-RNN-Hebb
Input weights	PC-RNN-HC-A
	PC-RNN-HC-M
	PC-RNN-HC-A-RS
	PC-RNN-HC-M-RS

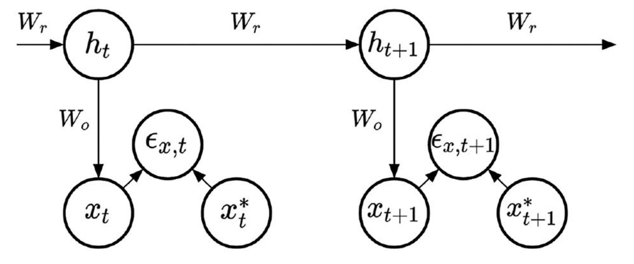


FIGURE 2 | Simple RNN model.

hidden states. Learning is then performed only on the synaptic weights involving the components of this subspace:

$$\boldsymbol{W}_o \leftarrow \boldsymbol{W}_o + \lambda \boldsymbol{\epsilon}_{x,t} \cdot \Big((\mathbb{I} - \boldsymbol{C}) \cdot \tanh(\boldsymbol{h}_t) \Big)^{\mathsf{T}} \tag{6}$$

As shown in Equation 4, low values of α induce a Conceptor matrix close to 0, leading to a projection matrix $(\mathbb{I} - C)$ close to the identity. On the opposite, high values of α induce a conceptor matrix close to the identity matrix, leading to a hard projection hindering learning.

The third learning mechanism that we study is the Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) algorithm applied to the output weights of the RNN. On each task k, we can compute the Fischer information matrix $\boldsymbol{F}_k$, where each coefficient $F_{k,i}$ measures the importance of the synaptic weight $W_{o,i}$:

$$F_{k,i} = \sum_{t=1}^{T} \left(\nabla_{W_{o,i}} \|\boldsymbol{x}_t - \boldsymbol{x}^*_{k,t}\|_2^2 \right)^2 \tag{7}$$

where $\boldsymbol{x}^*_{k,t}$ denotes the target at time t for the task k. Then, on a new task k', EWC minimizes the following loss function for each synaptic weight $W_{o,i}$:

$$\mathcal{L}(W_{o,i}) = \mathcal{L}_{k'}(W_{o,i}) + \sum_{k<k'} \frac{\beta}{2} F_{k,i}(W_{o,i} - W^*_{k,i})^2 \tag{8}$$

where $\mathcal{L}_{k'}$ denotes the loss for task k' without EWC regularization, β is a hyperparameter controlling the importance of the new task with regard to previous tasks, and $W^*_{k,i}$ denotes the i-th component of the optimal synaptic weights $\boldsymbol{W}^*_k$ learned on task k. We optimize this loss function using gradient descent on the synaptic weights $\boldsymbol{W}_o$, and obtain the following learning rule:

$$\begin{aligned}\boldsymbol{W_o} \leftarrow \boldsymbol{W_o} \; + \; & \lambda \boldsymbol{\epsilon}_{x,t} \cdot \tanh(\boldsymbol{h}_t)^{\mathsf{T}} \\ & -\lambda\beta \Big[\big(\sum_{k<k'} \boldsymbol{F_k} \big) \odot \boldsymbol{W_o} - \sum_{k<k'} \boldsymbol{F_k} \odot \boldsymbol{W}^*_k \Big]\end{aligned} \tag{9}$$

We can observe that the second line pulls $\boldsymbol{W_o}$ toward the optimal output weights found for previous tasks, weighted by coefficients measuring the importance of each synaptic weight. In terms of memory requirements, we need to store the sum of the Fischer matrices, as well as the sum of previous optimal synaptic weights weighted by the fisher matrices.

Finally, we also experiment with Generative Replay (GR) as a continual learning technique mitigating catastrophic forgetting. Since each individual task consists precisely in learning to generate the task data (the trajectory), the learned generative model can directly be used to provide samples of the previous tasks. We apply this technique to the simple ESN model described beforehand. At each new task k', we create a copy of the model trained on the tasks $k < k'$. This copy is used to generate samples $\{\boldsymbol{x}_1, \boldsymbol{x}_T\}$ that should be close to the previous tasks' trajectories. During training on the task k', the ESN is also trained in parallel to predict these replayed trajectories, which mitigates catastrophic forgetting.

3.2.2. Recurrent Weights

For the learning of the recurrent weights, we compare three learning methods inspired by PC. All three models share the same architecture, represented in **Figure 3**. On top of the top-down computations predicting the output $\boldsymbol{x}_t$, these models include bottom-up computations updating the value of the hidden state, and providing a prediction error signal on the hidden layer:

$$\boldsymbol{\epsilon}_{x,t} = \boldsymbol{x}^*_t - \boldsymbol{x}_t \tag{10}$$

$$\boldsymbol{h}^*_t = \boldsymbol{h}_t + \alpha_x \boldsymbol{W}_b \cdot \boldsymbol{\epsilon}_{x,t} \tag{11}$$

$$\boldsymbol{\epsilon}_{h,t} = \boldsymbol{h}^*_t - \boldsymbol{h}_t \tag{12}$$

where α_x is an update rate that weights the importance of bottom-up information for the estimation of $\boldsymbol{h}_t$.

In fact, the three models we compare propose the same update rule for the recurrent weights, they will only differ in their definition of the feedback weights, which impacts the recurrent weights update. The learning rule for the recurrent weights is based on the hidden state at time t and the prediction error on the hidden state layer at time $t+1$, according to the following equation:

$$\boldsymbol{W}_r \leftarrow \boldsymbol{W}_r + \lambda_r \boldsymbol{\epsilon}_{h,t+1} \cdot \tanh(\boldsymbol{h}^*_t)^{\mathsf{T}} \tag{13}$$

where λ_r is the learning rate of the recurrent weights.

The difference between the three models lies in the computation of $\boldsymbol{h}^*_{t+1}$. In the first model, that we label PC-RNN-V (for Vanilla), this bottom-up computation is done using the transposed of the top-down weights used for prediction. This results in a direct minimization of VFE, as shown in Appendix A. In the two other models, these feedback and bottom-up weights are instead learned. In the original PC model described in Rao and Ballard (1997), it was proposed to learn these feedback weights using the same rule as Equation 3 (up to a transpose to match the feedback weights shape):

$$\boldsymbol{W}_b \leftarrow \boldsymbol{W}_b + \lambda \tanh(\boldsymbol{h}_t) \cdot \boldsymbol{\epsilon}^{\mathsf{T}}_{x,t} \tag{14}$$

This learning rule ensures that with random initializations, but enough training time, the feedback weights converge to the transposed forward weights. Since this learning rule is a copy of the Hebbian rule used in Equation 3, we call PC-RNN-Hebb the

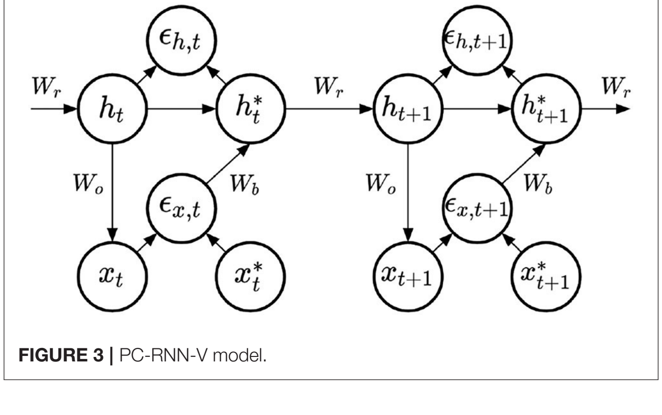


FIGURE 3 | PC-RNN-V model.

RNN model using this method. The last model, inspired by the P-TNCN (Ororbia et al., 2020), implements a different learning rule for the feedback weights, described by the following equation:

$$\boldsymbol{W}_b \leftarrow \boldsymbol{W}_b - \lambda_b \big(\boldsymbol{\epsilon}_{h,t} - \boldsymbol{\epsilon}_{h,t-1}\big) \cdot \boldsymbol{\epsilon}_{x,t}^{\intercal} \quad (15)$$

The model presented in Ororbia et al. (2020) also implements an additional term in the learning rule for the recurrent and output weights, on top of the rules explained here. This additional term led our experiments to worse results. For this reason, we do not provide more details about this rule and turn it off during the experiments shown below.

3.2.3. Input Weights

Finally, we compare four methods to learn RNN input weights. All methods share the same representation, displayed in **Figure 4**. This architecture was derived following the principle of free-energy minimization (Friston and Kilner, 2006), using a generative model that features a latent variable called hidden causes and labeled $\boldsymbol{c}$. Similarly to hidden states, hidden causes are hidden variables that can be dynamically inferred by the PC-RNN network. However, contrary to the hidden state variable, hidden causes are not dynamic: in the absence of prediction error the value of the hidden causes is stationary $\boldsymbol{c}_t = \boldsymbol{c}_0$. The derivations of these models are summarized in Appendix A. The resulting architecture takes as input an initial value for the hidden causes and predicts an output sequence while dynamically updating the hidden states and hidden causes. During training, this input is the one-hot encoded index of the current task $\boldsymbol{c}_0 = k$.

The four models differ according to two dimensions: whether they use evolution strategies to estimate the input weights, and according to the implementation of the influence of the input onto the hidden state dynamics. This influence can be either additive or multiplicative, the additive scheme is based on the following equation:

$$\boldsymbol{h}_t = (1 - \frac{1}{\tau})\boldsymbol{h}_{t-1}^{*} + \frac{1}{\tau}\big(\boldsymbol{W}_r \cdot \tanh(\boldsymbol{h}_{t-1}^{*}) + \boldsymbol{W}_i \cdot \boldsymbol{c}_{t-1}\big) \quad (16)$$

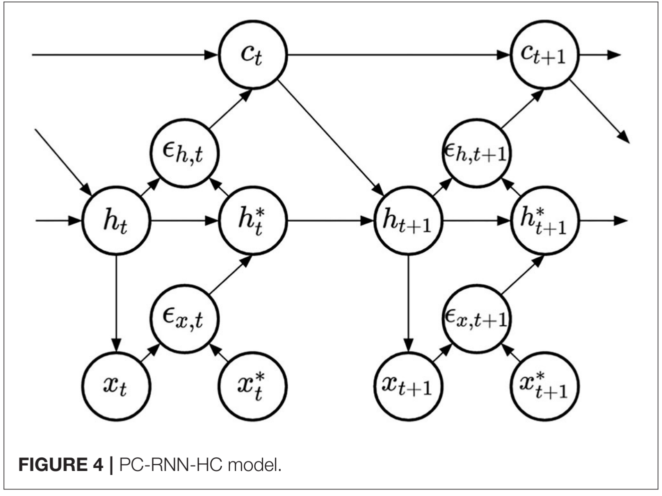


FIGURE 4 | PC-RNN-HC model.

The multiplicative scheme is based on the following equation:

$$\begin{aligned}\boldsymbol{h}_t &= (1 - \frac{1}{\tau})\boldsymbol{h}_{t-1}^{*} \\ &+ \frac{1}{\tau}\boldsymbol{W}_f^{\intercal} \cdot \Big(\big(\boldsymbol{W}_p \cdot \tanh(\boldsymbol{h}_{t-1}^{*})\big) \odot (\boldsymbol{W}_i \cdot \boldsymbol{c}_{t-1})\Big) \quad (17)\end{aligned}$$

where we have introduced new synaptic weights $\boldsymbol{W}_p$ and $\boldsymbol{W}_f$, that replace the recurrent weights of the additive version. This reparameterization is used to reduce the total number of parameters of the multiplicative RNN, as already used in Annabi et al. (2021a,b).

We label these two models, respectively, PC-RNN-HC-A and PC-RNN-HC-M, the HC suffix standing for Hidden Causes and the A and M suffixes standing for Additive and Multiplicative. The differences between the additive and multiplicative models also impact the bottom-up update rule for $\boldsymbol{c}_t$. However, in our experiments, we always turn off this mechanism by using an update rate equal to zero.

In these two first methods, the learning rules for the input weights follow the PC theory and attempt at minimizing the prediction error on the hidden layer. The learning rule used for the PC-RNN-HC-A model is the following:

$$\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i + \lambda_i \boldsymbol{\epsilon}_{h,t+1} \cdot \boldsymbol{c}_t^{\intercal} \quad (18)$$

For the PC-RNN-HC-M model, we obtain the following rule:

$$\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i + \lambda_i \Big(\big(\boldsymbol{W}_p \cdot \tanh(\boldsymbol{h}_t^{*})\big) \odot \big(\boldsymbol{W}_f \cdot \boldsymbol{\epsilon}_{h,t+1}\big)\Big) \cdot \boldsymbol{c}_t^{\intercal} \quad (19)$$

The third and fourth methods that we study are respectively based on the PC-RNN-HC-A and PC-RNN-HC-M, but instead use random search to optimize the weights $\boldsymbol{W}_i$. Our implementation of this random search is inspired by the learning algorithm proposed in Pitti et al. (2017):

$$\boldsymbol{\delta}_i \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{d_h^2}) \quad (20)$$

$$\|\boldsymbol{\epsilon}_{x,i}\|_2 \leftarrow \text{simulate}(\boldsymbol{W}_i + \boldsymbol{\delta}_i) \quad (21)$$

$$\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i + \delta_i sign(\|\boldsymbol{\epsilon}_{x,i-1}\|_2 - \|\boldsymbol{\epsilon}_{x,i}\|_2) \quad (22)$$

where the function *sign* associates -1 to negative values and 1 to positive values. At each training iteration i, the algorithm samples a noise matrix $\boldsymbol{\delta}_i$ that is added to the input weights of the RNN. After generation, the difference between the old and new average norm of the prediction error $\|\boldsymbol{\epsilon}_{x,i-1}\|_2 - \|\boldsymbol{\epsilon}_{x,i}\|_2$ is used as a measure of success of the addition of $\boldsymbol{\delta}_i$ and weights the update of $\boldsymbol{W}_i$. Since this algorithm only relies on an average of the prediction error over the predicted sequences, that can be computed iteratively, it qualifies as an online learning algorithm.

In summary, we have identified four learning algorithms for output weights, three learning algorithms for recurrent weights, and four learning algorithms for input weights. To connect the proposed methods to the classification of continual learning methods presented above, we could categorize the Conceptors method as a regularization method, and the fact that new tasks are associated with new inputs to the RNN in the shape of hidden causes, as an architecture modification method.

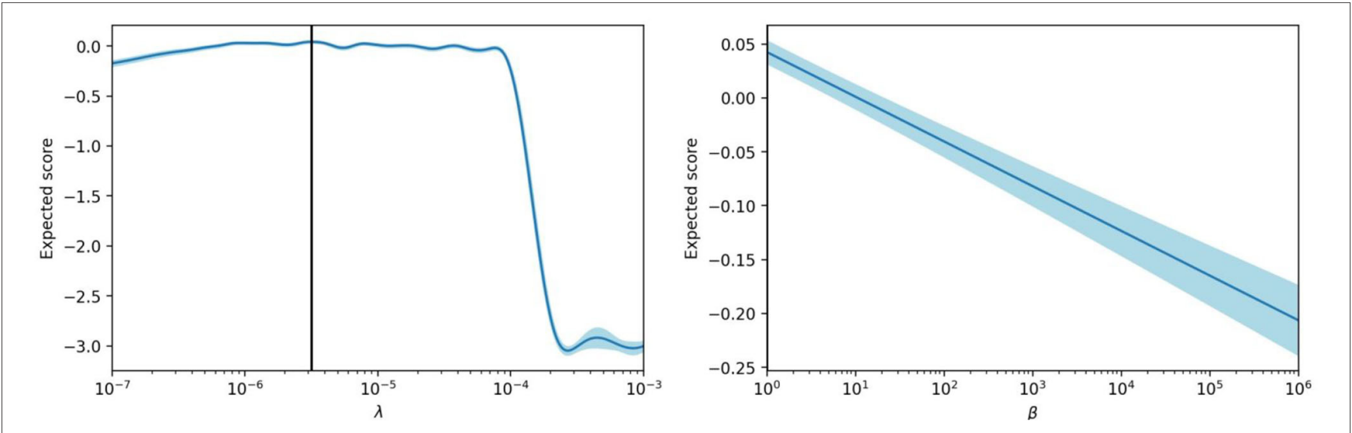


FIGURE 5 | Score estimation of the hyperparameter optimizer with regard to the learning rate λ and the coefficient β, for the EWC model.

4. RESULTS

4.1. Hyperparameter Optimization

The source code for the experiments presented in this section is available on GitHub[1]. It contains our implementation of the different models as well as the hyperparameter optimization method. In Appendix B, we provide the optimal hyperparameters found for each model.

We start by showing an example of a hyperparameter optimization in **Figure 5**, which was performed on the EWC model with $d_h = 300$. The optimized hyperparameters are the learning rate of the output weights, λ, and the coefficient β. After trying 200 hyperparameter configurations, the optimizer can estimate the score for all the configurations within the given range of values. These figures display the evolution of the score estimation according to λ using the optimal value for β, and according to β using the optimal value for λ. We can see that the function according to β monotonically decreases, while the function according to λ increases steadily before dropping once we attain values of the learning rate that no longer sustain convergence of the gradient descent.

In this case, the hyperparameter optimization has found that the EWC regularization does not improve the final score, and suggests using the lowest possible value for the coefficient β. When β increases, the regularization mitigates catastrophic forgetting but prevents proper learning of new tasks.

For all the results presented below, we perform optimization of the hyperparameters following the same protocol.

4.2. Output Weights

In **Figure 6**, we represent the average prediction error over 10 seeds for the continual learning of 20 sequential patterns obtained on the test set, with the hyperparameters found using the protocol described before. The vertical dashed lines in these figures delimit each of the training tasks. The colored lines represent the individual prediction error for each of the 20 sequence patterns (averaged over the 10 seeds). Finally, the black line represents the average prediction error over all the sequence patterns (averaged over the 10 seeds).

During each task (for each colored line), we can observe that one of the individual prediction errors decreases rapidly, while the other prediction errors only slightly change. Once the training task corresponding to a certain sequence pattern k is over, the prediction error associated with this pattern tends to increase. The better learning mechanism is the one that can limit this undesirable forgetting of previously learned sequence patterns. We can observe in **Figure 6** the Conceptors learning mechanism limits forgetting compared to the standard stochastic gradient descent rule used in our ESN model.

At first, it can be surprising that for each individual task, the corresponding prediction error reaches a lower value for the Conceptors model than for the ESN model. In terms of learning rules, the ESN model could potentially learn each pattern with better accuracy by increasing the learning rate. However, the hyperparameter optimizer has estimated that an increased learning rate would be detrimental to the complete continual learning task. Indeed, increasing the learning rate might improve the learning on every individual task, but it would also lead to more forgetting throughout the complete task. It is only because the Conceptors learning mechanisms naturally limit forgetting that the hyperparameter optimized "allows" a higher learning rate and, thus, better learning on each individual task.

We can also observe that the prediction error level that is reached during each individual task using the Conceptors model seems to increase throughout the complete task. We suppose that this is a consequence of further learning being prevented on synaptic connections associated with previous tasks' associated Conceptors. When a large number of individual tasks are over, learning is limited to synapses corresponding to a subspace of the hidden state space not belonging to any of the previous Conceptors. Decreasing the aperture α would allow better learning of the late tasks, but at the detriment of an increased forgetting of the early tasks.

[1]https://github.com/sino7/continual_online_learning_rnn_benchmark

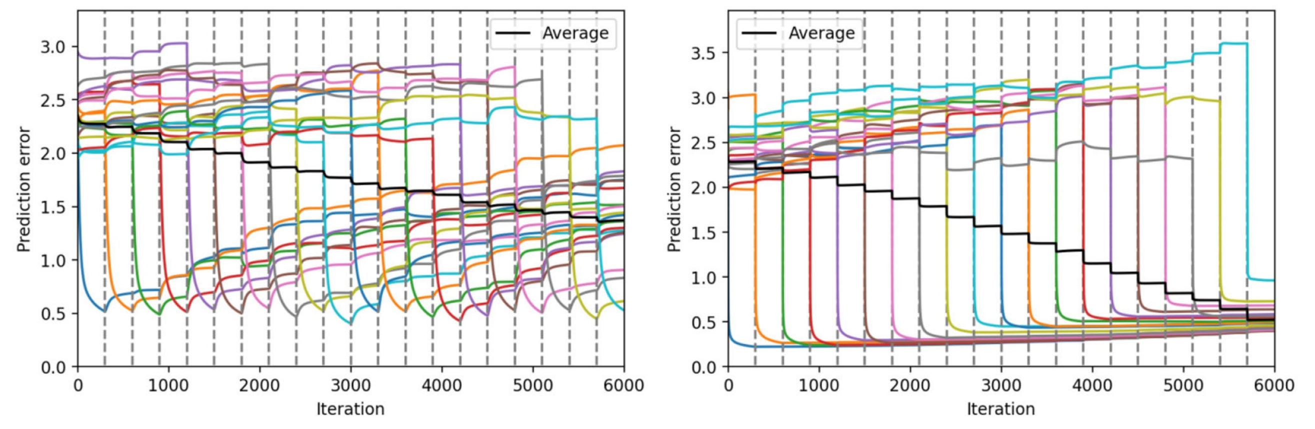


FIGURE 6 | Continual learning results with the ESN model (left) and the Conceptors model (right). We represent the average prediction error over 10 seeds, for the continual learning of 20 sequential patterns, obtained on the first test set. The colored lines correspond to the prediction error on each individual task, and the black line corresponds to the prediction error averaged on all tasks. The 20 tasks are delimited by the dashed gray lines.

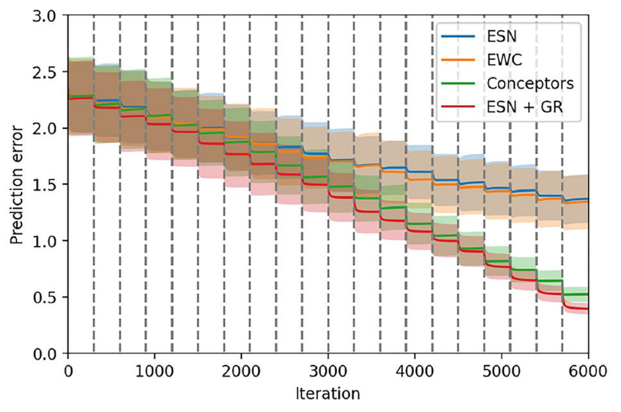


FIGURE 7 | Comparison between the four learning methods for the output weights on the first test set. The 20 tasks are delimited by the dashed gray lines.

TABLE 2 | Average prediction error after training on all p tasks.

Model	Validation (MOCAP, $p = 15$)	Test 1 (MOCAP, $p = 20$)	Test 2 (Handwriting, $p = 20$)
ESN	0.90 ± 0.07	1.37 ± 0.14	0.71 ± 0.04
EWC	0.90 ± 0.09	1.35 ± 0.15	0.69 ± 0.05
Conceptors	0.31 ± 0.02	0.52 ± 0.04	0.27 ± 0.02
ESN + GR	**0.29 ± 0.01**	**0.39 ± 0.01**	**0.22 ± 0.01**
PC-RNN-V	0.87 ± 0.09	1.41 ± 0.14	0.79 ± 0.10
P-TNCN	0.90 ± 0.08	1.42 ± 0.18	0.71 ± 0.05
PC-RNN-Hebb	0.90 ± 0.07	1.41 ± 0.10	0.73 ± 0.05
PC-RNN-HC-A	**0.74 ± 0.09**	**1.28 ± 0.22**	**0.59 ± 0.04**
PC-RNN-HC-M	0.81 ± 0.04	1.32 ± 0.09	0.77 ± 0.05
PC-RNN-HC-A-RS	0.90 ± 0.08	1.39 ± 0.15	0.77 ± 0.05
PC-RNN-HC-M-RS	0.93 ± 0.06	1.38 ± 0.10	0.72 ± 0.05
PC-Conceptors	**0.28 ± 0.01**	**0.36 ± 0.02**	**0.18 ± 0.01**

Bold value indicates the best performance in each group of models.

Figure 7 compiles these previous figures to compare the average prediction error using the four learning mechanisms for output weights. At the end of the training, we can see that the Conceptors model and generative replay achieve a significantly lower prediction error than the ESN using the standard stochastic gradient descent rule and the EWC regularization for the learning of the output weights.

As explained in the last section, the hyperparameters found for EWC correspond to a configuration where the regularization is almost removed, and the EWC model, thus, has the same performance as the ESN model.

The generative replay strategy outperforms all other approaches, but at the cost of a longer training time. Indeed, at each task k, the model is trained on $(k - 1)$ replayed trajectories on top of the current trajectory. For all models, we have limited the number of training iterations on each task, which induces an unfair advantage for generative replay in our experiments. For this reason, we do not include this technique in the remaining comparisons.

The results obtained with these models on the three data sets (validation set and two test sets) are provided in **Table 2**, together with the results for the learning of recurrent and input weights, discussed in the next sections.

4.3. Recurrent Weights

In this second experiment, we compare the PC-RNN-V with two variants using learning rules for the feedback weights instead of using the transposed feedforward weights. These three learning methods in the end provide different update rules for the recurrent weights of the RNN. The results of this second comparative analysis are provided in **Table 2**.

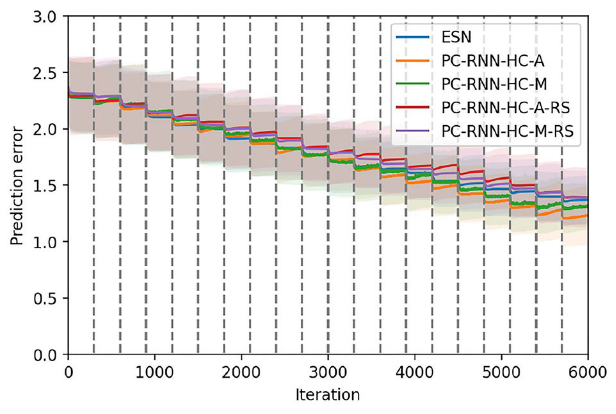


FIGURE 8 | Comparison between the three learning methods for the input weights. The PC-RNN-V model, where no learning is performed on the input weights, is also displayed as a baseline. The 20 tasks are delimited by the dashed gray lines.

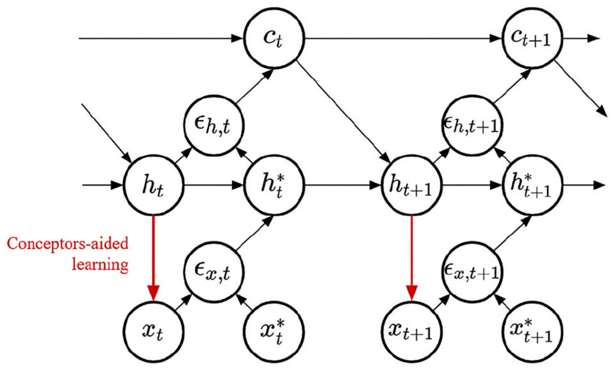


FIGURE 9 | PC-Conceptors model.

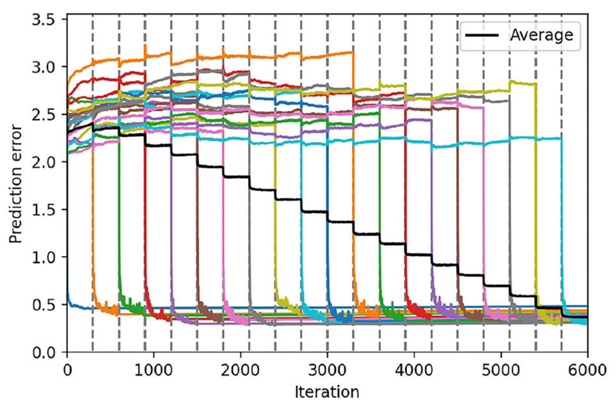


FIGURE 10 | Continual learning results using the PC-Conceptors. We represent the average prediction error over 10 seeds, for the continual learning of 20 sequential patterns, using the PC-RNN-HC-A model with Conceptors. The colored lines correspond to the prediction error on each individual task, and the black line corresponds to the prediction error averaged on all tasks. The 20 tasks are delimited by the dashed gray lines.

We can see that none of the three models brings any significant improvement compared with the ESN, which is exactly the same model without any learning occurring on the recurrent weights. In terms of hyperparameters, only the PC-RNN-V has an optimal learning rate for recurrent weights that does not correspond to the lowest value authorized during hyperparameter optimization. This means that for both P-TNCN and PC-RNN-Hebb models, the hyperparameter optimizer has estimated that training the recurrent weights only hinders the final prediction error. For the PC-RNN-V model, a slight improvement was found in the validation set using the learning rule for recurrent weights, but this improvement does not transfer to the two test sets.

We can conclude based on these results that recurrent weights learning in a continual learning setting is difficult and might often lead to more catastrophic forgetting.

4.4. Input Weights

Figure 8 displays the results obtained with the four learning mechanisms for input weights, and the ESN as a baseline. We use the ESN as a baseline to measure the improvement brought by the learning in the input layer. The results of the validation set and other test sets are displayed in **Table 2**.

These results suggest that the learning methods using random search (RS suffix) perform poorly compared to the corresponding learning rules relying on the propagation of error using PC. The two models using random search perform similarly to the baseline ESN model. This observation is surprising since the $\boldsymbol{W}_i$ weights in PC-RNN-HC-A/M architectures are directly factored according to each individual task. Indeed, during the task k, we can limit learning on the k-th column of the $\boldsymbol{W}_i$ weights, since these are the only weights that influence the RNN trajectory. Consequently, training this layer should not cause any additional forgetting, and thus should only bring improvements over the baseline ESN model. Since the two models using random search did not bring any improvement, we suppose that this is due to the limited number of iterations allowed for the training on each individual task. We observed that in general training with random search as in the INFERNO model (Pitti et al., 2017) needed many more iterations than gradient-based methods.

The PC-RNN-HC-A/M models trained using the PC-based learning rules still showed some significant improvement compared with the ESN baseline, with the PC-RNN-HC-A model performing slightly better than the PC-RNN-HC-M model. This experiment allows us to conclude that the learning rule for input weights proposed by the PC-RNN-HC-A model is the most suited to a continual learning setting.

4.5. Combining Conceptors and Hidden Causes

Finally, we can inquire whether these different learning mechanisms combine well with each other. We implement the Conceptors learning rule on the output weights of a PC-RNN-HC-A model, a new model that we label PC-Conceptors, as represented in **Figure 9**. **Figure 10** displays the prediction error on each individual task as well as the average prediction error throughout learning, using this model. Interestingly,

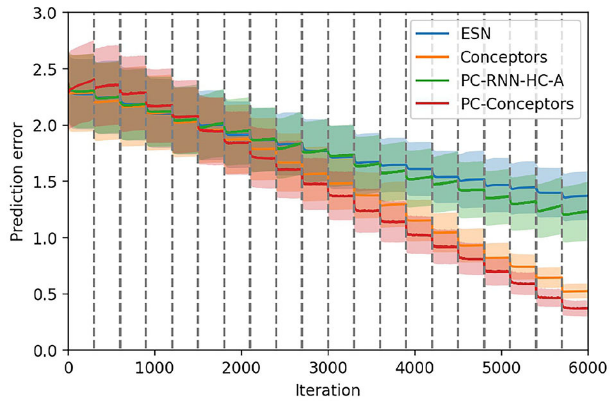


FIGURE 11 | Comparison between the ESN, the Conceptors model, the PC-RNN-HC-A model, and the PC-Conceptors model on the first test set. The 20 tasks are delimited by the dashed gray lines.

virtually no forgetting seems to happen during learning, as the individual prediction errors plateau after decreasing during the corresponding individual tasks.

Additionally, the hyperparameter optimizer in this case recommended using the lowest possible value for the recurrent weights learning rate. This suggests that the recurrent weights learning negatively interferes with the Conceptors model. The Conceptors model might be sensible for recurrent weight learning, since this could turn the previously learned Conceptors into obsolete descriptors of the corresponding hidden state trajectories.

We compare these results with the ESN, Conceptors and PC-RNN-HC-A models in **Figure 11**, which confirms that this combination of learning methods seems to provide the RNN model best suited for online continual learning.

5. DISCUSSION

Overall, this study suggests that regularization methods such as Conceptors, and architectural methods, as proposed in the PC-RNN-HC architectures, can help design RNN models with online learning rules suitable for continual learning.

Additionally, we have found that combining Conceptors-based learning for the output weights with PC-based learning for the input weights further improves the model precision. In future study, it would be interesting to investigate whether the combination of these two mechanisms could be improved. Especially, the learning of the input weights is only driven by the minimization of the prediction error on the recurrent layer. This could be improved by integrating an orthogonality criterion to the learning rule: if the input weights are optimized in order to decorrelate the different hidden state trajectories, it could facilitate the learning of the output weights.

The models we have proposed also suffer from another limitation that should be addressed in future work. The models were trained using as input the current task index, which is information that might not be available in realistic lifelong learning settings. The model should be able to detect a distributional shift when it occurs and adapt its learning rules based on these events.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found in online repositories. The name of the repository and accession number can be found below: GitHub, https://github.com/sino7/continual_online_learning_rnn_benchmark.

AUTHOR CONTRIBUTIONS

The models and experiments were designed by LA, AP, and MQ. The models and experiments were implemented by LA. The article was written by LA with instructions and feedback from AP and MQ. All authors contributed to the article and approved the submitted version.

FUNDING

This study was funded by the Cergy-Paris University Foundation (Facebook grant) and partially by Labex MME-DII, France (ANR11-LBX-0023-01).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot.2022.845955/full#supplementary-material

REFERENCES

Annabi, L., Pitti, A., and Quoy, M. (2021a). Bidirectional interaction between visual and motor generative models using predictive coding and active inference. *Neural Netw*. 143, 638–656. doi: 10.1016/j.neunet.2021.07.016

Annabi, L., Pitti, A., and Quoy, M. (2021b). "A predictive coding account for chaotic itinerancy," in *Artificial Neural Networks and Machine Learning-ICANN 2021*, eds I. Farkaš, P. Masulli, S. Otte, and S. Wermter (Cham: Springer International Publishing), 581–592.

Clark, A. (2013). Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behav. Brain Sci*. 36, 181–204. doi: 10.1017/S0140525X12000477

Collins, J., Sohl-Dickstein, J., and Sussillo, D. (2016). Capacity and trainability in recurrent neural networks. *stat* 1050:29.

Cossu, A., Bacciu, D., Carta, A., Gallicchio, C., and Lomonaco, V. (2021a). Continual learning with echo state networks. *arXiv preprint* arXiv:2105.07674. doi: 10.14428/esann/2021.ES2021-80

Cossu, A., Carta, A., Lomonaco, V., and Bacciu, D. (2021b). Continual learning for recurrent neural networks: an empirical evaluation. *Neural Netw*. 143, 607–627. doi: 10.1016/j.neunet.2021.07.021

Dua, D., and Graff, C. (2019). *Uci machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Science. Available online at: http://archive.ics.uci.edu/ml.

Friston, K., and Kilner, J. (2006). A free energy principle for the brain. *J. Physiol. Paris* 100:70–87. doi: 10.1016/j.jphysparis.2006.10.001

Jaeger, H. (2001). *The âĂIJecho stateâĂİ approach to analysing and training recurrent neural networks*. GMD-Report 148, German National Research Institute for Computer Science.

Jaeger, H. (2014a). Conceptors: an easy introduction. *CoRR abs/1406.2671*.

Jaeger, H. (2014b). Controlling recurrent neural networks by conceptors. *CoRR abs/1403.3369*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. U.S.A*. 114, 3521–3526. doi: 10.1073/pnas.1611835114

Li, Z., and Hoiem, D. (2017). Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell*. 40, 2935–2947. doi: 10.1109/TPAMI.2017.2773081

Lukosevicius, M., and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev*. 3, 127–149. doi: 10.1016/j.cosrev.2009.03.005

Maass, W., NatschlÃď'ger, 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

Mallya, A., Davis, D., and Lazebnik, S. (2018). "Piggyback: adapting a single network to multiple tasks by learning to mask weights," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, 67–82.

McCloskey, M., and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: the sequential learning problem. *Psychol. Learn. Motivat*. 24, 109–165. doi: 10.1016/S0079-7421(08)60536-8

Millidge, B., Tschantz, A., and Buckley, C. L. (2020). Predictive coding approximates backprop along arbitrary computation graphs. *CoRR, abs/2006.04182*.

Ororbia, A., Mali, A., Giles, C. L., and Kifer, D. (2020). Continual learning of recurrent neural networks by locally aligning distributed representations. *IEEE Trans. Neural Netw. Learn. Syst*. 31, 4267–4278. doi: 10.1109/TNNLS.2019.2953622

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13* (JMLR.org), III–1310–III–1318.

Pitti, A., Gaussier, P., and Quoy, M. (2017). Iterative free-energy optimization for recurrent neural networks (inferno). *PLoS ONE* 12, e0173684. doi: 10.1371/journal.pone.0173684

Rao, R., and Ballard, D. (1999). Predictive coding in the visual cortex a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci*. 2, 79–87. doi: 10.1038/4580

Rao, R. P. N., and Ballard, D. H. (1997). Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Comput*. 9, 721–763. doi: 10.1162/neco.1997.9.4.721

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). "ICARL: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (Honolulu, HI: IEEE), 2001–2010.

Schmidhuber, J., Wierstra, D., Gagliolo, M., and Gomez, F. (2007). Training recurrent networks by evolino. *Neural Comput*. 19, 757–779. doi: 10.1162/neco.2007.19.3.757

Schmidhuber, J., Wierstra, D., and Gomez, F. (2005). "Evolino: hybrid neuroevolution / optimal linear search for sequence learning," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI'05* (San Francisco, CA: Morgan Kaufmann Publishers Inc.), 853–858.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. *arXiv preprint* arXiv:1705.08690. doi: 10.48550/arXiv.1705.08690

Sodhani, S., Chandar, S., and Bengio, Y. (2020). Toward training recurrent neural networks for lifelong learning. *Neural Comput*. 32, 1–35. doi: 10.1162/neco_a_01246

Sussillo, D., and Abbott, L. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron* 63, 544–557. doi: 10.1016/j.neuron.2009.07.018

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

Whittington, J. C. R., and Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Comput*. 29, 1229–1262. doi: 10.1162/NECO_a_00949

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

frontiers | Frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 25 May 2022
doi: 10.3389/fnbot.2022.882518

Design Principles for Neurorobotics

Jeffrey L. Krichmar[1,2*†] *and Tiffany J. Hwu*[3†]

[1] Department of Cognitive Sciences, University of California, Irvine, Irvine, CA, United States, [2] Department of Computer Science, University of California, Irvine, Irvine, CA, United States, [3] Riot Games, Los Angeles, CA, United States

In their book "How the Body Shapes the Way We Think: A New View of Intelligence," Pfeifer and Bongard put forth an embodied approach to cognition. Because of this position, many of their robot examples demonstrated "intelligent" behavior despite limited neural processing. It is our belief that neurorobots should attempt to follow many of these principles. In this article, we discuss a number of principles to consider when designing neurorobots and experiments using robots to test brain theories. These principles are strongly inspired by Pfeifer and Bongard, but build on their design principles by grounding them in neuroscience and by adding principles based on neuroscience research. Our design principles fall into three categories. First, organisms must react quickly and appropriately to events. Second, organisms must have the ability to learn and remember over their lifetimes. Third, organisms must weigh options that are crucial for survival. We believe that by following these design principles a robot's behavior will be more naturalistic and more successful.

Keywords: adaptive behavior, embodiment, learning, memory, neuromodulation, value

OPEN ACCESS

Edited by:
Florian Röhrbein,
Technische Universität Chemnitz,
Germany

Reviewed by:
André Cyr,
University of Ottawa, Canada
Hendry Ferreira Chame,
Université de Lorraine, France

****Correspondence:***
Jeffrey L. Krichmar
jkrichma@uci.edu

†These authors have contributed equally to this work

Received: *23 February 2022*
Accepted: *19 April 2022*
Published: *25 May 2022*

Citation:
Krichmar JL and Hwu TJ (2022) Design Principles for Neurorobotics. Front. Neurorobot. 16:882518. doi: 10.3389/fnbot.2022.882518

1. INTRODUCTION

Neurorobotics is a powerful tool for testing brain theories and increasing our understanding of neuroscience. The robot controller is modeled after some aspect of the nervous system. Unlike human or other animal studies, the neuroroboticist has access to every aspect of this artificial brain during the lifetime of the agent. Therefore, the neuroroboticist can analyze and perturb the nervous system in ways that a neuroscientist cannot with present recording technology. Not only can a neurorobot be tested under laboratory conditions that are similar to those of an animal experiment in order to provide direct comparisons, but it can also be tested in more natural conditions to see how these brain functions might respond to real-world situations.

The actions of a neurorobot are embedded in its environment. By choosing an appropriate morphology, simple mechanical designs can perform complex functions by taking advantage of environmental features, thus alleviating slow, power-hungry nervous systems from having to make these calculations. This is known as morphological computation (Pfeifer and Bongard, 2006). Neurorobot designs can be degenerate, that is, they can contain multiple systems capable of performing the same functions (Edelman and Gally, 2001). In this way the agent can still survive in the environment should one system fail. Similar to many mobile operating systems, neurorobot computation can follow the brain architecture by having multiple processes run in parallel in an event-driven manner, continuously responding to concurrent events. These ideas have roots in behavior-based robots (Brooks, 1991; Arkin, 1998) and the design of neuromorphic hardware (Merolla et al., 2014; Davies et al., 2018).

To adapt to a changing environment a neurorobot must be able to learn, store and recall information. Memory systems in neurorobotics are particularly applicable in spatial memory for navigation and contextual memory for learning representations of the environment (Milford et al., 2016; Gaussier et al., 2019; Hwu et al., 2020). Success within dynamic environments, such as the real-world, requires the processing of risk, reward, and uncertainty by some notion of value and the ability to adapt (Oudeyer and Kaplan, 2007; Krichmar, 2008; Merrick, 2017). Through such systems, neurorobots are able to predict future events and adapt to changes in the environment.

The world is full of trade-offs and changing needs that require us to make choices. Incorporating behavioral trade-offs into neurorobots such as reward vs. punishment, invigorated vs. withdrawn activity, expected vs. unexpected uncertainty for attention, exploration vs. exploitation of choices, foraging for food vs. defending one's territory, coping with stress vs. keeping calm, and social interaction vs. solitary restraint can lead to interesting behavior in neurorobotics (Canamero et al., 2006; Hiolle et al., 2012; Krichmar, 2013; Lones et al., 2018). Many of these trade-offs are regulated by the neuromodulators and hormone levels in the brain.

In this article, we present a set of principles to take into consideration when designing neurorobots. They fall into three categories: 1) Embodiment and reactions, 2) Adaptive behavior, learning and memory, and 3) Behavioral trade-offs. Following these design principles can make neurorobots more naturalistic and more interesting. Many of the ideas put forth in this article are based on material from our forthcoming book (Hwu and Krichmar, 2022).

2. EMBODIMENT AND REACTIONS - RESPONDING TO THE HERE AND NOW

In this first set of neurorobotic design principles, we focus on what (Pfeifer and Bongard, 2006) called the "here and now." These design principles are grounded in neuroscience and are focused on processes that respond to events. Even without learning and memory, these processes lead to flexible, adaptive behavior.

2.1. Embodiment

Brains do not work in isolation; they are closely coupled with the body acting in its environment (Chiel and Beer, 1997). Biological organisms perform *morphological computation*; that is, certain processes are performed by the body that would otherwise be performed by the brain (Pfeifer and Bongard, 2006). Moment-to-moment action can be handled at the periphery by the body, sensors, actuators, and reflexes at the spinal cord level. This allows the central nervous system, which is slower and requires more processing than the body or peripheral nervous system, to predict, plan, and adapt by comparing its internal models with current information from the body (Shadmehr and Krakauer, 2008; Hickok et al., 2011).

In biology, the morphology and behavior must fit within the organism's ecological niche. Therefore, the layout of its sensors and actuators, their resolution and range are tuned to meet the specific organism's needs (Ziemke, 2003). As an example, consider a toddler flailing his or her arms. The child's arms more easily move toward the front of the torso than the back. By chance the toddler's hand touches an object causing a reflexive grasping motion. This leads to the fingers, where most of the tactile sensors are located, touching the object. The child will then move this hand in its easiest direction, which tends to be toward the face, where a range of sensors for vision, olfaction, and taste receptors reside. Comparing that with the embodiment and design of an insect or a fish, it is clear that these design implementations are specific to the organism's niche. Attention to these environmental details can provide guidance for the design of neurorobots. The form of the organism's body shapes its behavior and its brain function. This requires a brain and body that is engaged with the environment, which is what we should strive for in designing our neurorobots.

For many tasks that we carry out with ease, our brains are too slow to sense, process and move. For example, skiing down a hill or catching a wave on a surfboard happens too fast for our central nervous system to position the body appropriately. But the form and compliance of the body can position itself properly and adjust itself to perturbations without brain control. This is morphological computation in action.

Trapping a soccer ball is another example of morphological computation. In a RoboCup tournament, the Segway soccer team from The Neurosciences Institute solved this difficult sensorimotor problem with a very cheap design (Fleischer et al., 2006). On a large playing field it was nearly impossible for the robot to catch a fast-moving soccer ball, given that the Segway was large and cumbersome and had a slow camera frame rate and slow IR sensor refresh rate. Soccer balls would bounce off the robot before it had a chance to respond. After much trial and error, the team used plastic tubing that was fastened around the robot's body like a hula-hoop at just the right height (see **Figure 1**). Any ball that was passed to the Segway robot was trapped by the tubing, giving the robot time to use its camera and IR proximity sensors to place the ball in its kicking apparatus. In a sense, this is what humans do when playing soccer. They use soft pliant materials angled appropriately to soften the impact of a ball coming toward them. Many actions like these take place without much thought (i.e., brain processing).

By putting more emphasis on designs that exploit the environment, we can offload some of the control from the cognitive robot's central nervous system onto the body itself. This should allow the robot to be more responsive to the environment and more fluid in its actions. In addition, it frees up the nervous system to put more emphasis on planning, prediction, and error correction rather than reflexive movements. Too often cognitive neuroscientists forget that the body and the peripheral nervous system are performing many vital, moment-to-moment behaviors and tasks without central control. Even functions that are thought to be purely mental have a basis in embodiment. For example, it has been argued that interactions in which a person needs to understand another is an embodied process rather than an internal simulation (Gallagher, 2001).

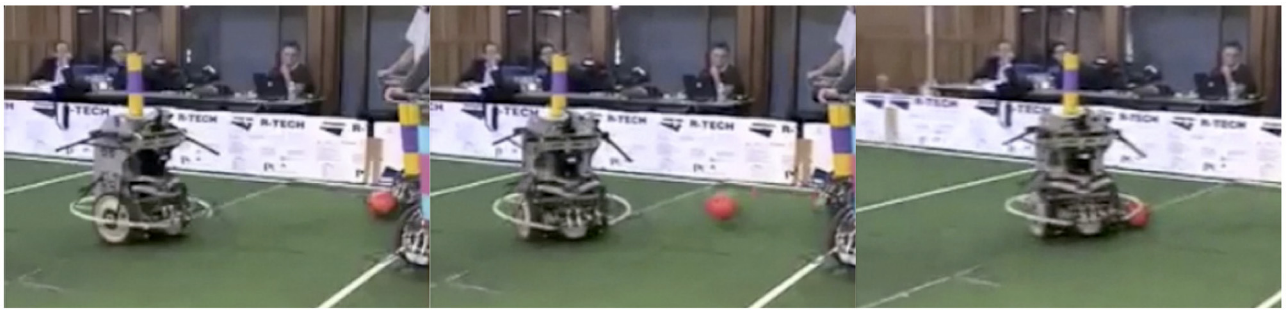


FIGURE 1 | Video sequence showing robot capturing a soccer ball using morphological computation. A pliant plastic hoop around the Segway robot allowed trapped the soccer ball, allowing the slower IR sensors and camera to confirm that a soccer ball was caught.

2.2. Efficiency Through Cheap Design

Cheap design means finding the simplest solution to the challenge the robot is facing. One way to do this is by exploiting the environment. For example, winged insects and fast swimming fish exploit their environment by creating vortices with their wing beats or fin movements, which causes additional thrust and more energy to come out than the animal put in. Most of our touch receptors are where we need them most, at our fingertips. It would be wasteful to have this fine level of resolution on the back of our hands or arms. This is what is meant by cheap.

For example, in legged locomotion, roboticists have put much time and effort in creating robust controllers for legged locomotion. The importance of the cheap design principle can be observed when comparing the biped locomotion of passive dynamic walking robots to sophisticated humanoid robots, such as Honda's Asimo or Aldebaran's NAO. Passive dynamic walking robots exploit gravity, friction, and the forces generated by their swinging limbs (Collins et al., 2005). As a result, they require very little energy or control to move (see **Table 1**). In contrast, robots such as Asimo need complex control systems and long-lasting batteries to achieve the same result. Although these passive walkers are not necessarily biologically inspired, once the engineers or artists implement a design that minimizes energy expenditure, the gait looks very natural.

However, it should be stressed that passive dynamics and morphological computation are not enough to support a complete range of natural behaviors. Rather it frees up the system from expending energy and computational resources on some functions, while allowing it to concentrate on other functionality. Saving energy is a recurring theme in biology since biological organisms are under tight metabolic constraints (Beyeler et al., 2019; Krichmar et al., 2019). However, there is a trade-off that comes with efficiency. For example, it is more efficient to walk on four legs, but then arms are not available for manual dexterity and gestures, which is important for bipedal organisms. In a complete system, passive control is closely coupled with spinal cord reflexes, which in turn are in close communication with motor cortex and other areas of the brain. These issues should be taken into consideration when designing neurorobots.

It is not only the body that follows the principle of cheap design: brains do as well. Biological systems are under extreme metabolic constraints and need to represent information efficiently. Therefore, the nervous system must encode information as cheaply as possible. The brain operates on a mere 20 watts of power, approximately the same power required for a ceiling fan operating at low speed (Krichmar et al., 2019). Although being severely metabolically constrained is at one level a disadvantage, evolution has optimized brains in ways that lead to incredibly efficient representations of important environmental features that are distinctly different from those employed in current digital computers. The brain utilizes many means to reduce its functional metabolic energy use. Indeed, one can observe at every level of the nervous system strategies to maintain high performance and information transfer while minimizing energy expenditure.

TABLE 1 | Energy consumed during legged locomotion (Collins et al., 2005). Unit weight per unit distance.

Agent	Energy consumption
Asimo	3.23
Cornell biped	0.20
Humans	0.20

At the neuronal coding level, the brain uses several strategies to reduce neural activity without sacrificing performance. Neural activity (i.e., the generation of an action potential, the return to resting state, and synaptic processing) is energetically very costly, and this drives the minimization of the number of spikes necessary to encode the neural representation of a new stimulus. Such sparse coding strategies appear to be ubiquitous throughout the brain (Olshausen and Field, 2004; Beyeler et al., 2019). Efficient coding reduces redundancies and rapidly adapts to changes in the environment. At a macroscopic scale, the brain saves energy by minimizing the wiring between neurons and brain regions (i.e., number of axons) and yet still communicates information at a high level of performance (Laughlin and Sejnowski, 2003). Information transfer between neurons and brain areas is preserved by a small-world network architecture, which reduces signal propagation (Sporns, 2010). These energy-saving ideas should be taken into consideration in constructing neural controllers for robots, which like biological organisms have limited energy resources. Moreover, many of these strategies could inspire new methodologies for constructing power-efficient artificial intelligence.

2.3. Sensory-Motor Integration

In the brain, the sensory and motor systems are tightly coupled. An organism or robot may get new sensory information that causes an action. Each action then creates new sensory information. Neurorobots can take advantage of this tightly coupled loop. For example, figure-ground segregation is a difficult computer vision problem in which a scene of static objects needs to be recognized from the background (e.g., a small toy block sitting on a similarly colored table). However, segmentation can be facilitated by sensorimotor integration in a very natural way as was demonstrated in a robot experiment by Fitzpatrick and Metta (2003). In their study, the robot's hand moved until it happened to hit the toy block, triggering motion detector responses in its visual system. In this way the robot's motor system generated sensory information, both visual and tactile, which led to the unexpected recognition of an object on the table. In the nervous system, a copy of the action, called a motor efference copy, is fed back to the brain. It creates an expectation that can be used to error check the movement and the expected sensory experience. Because hitting the toy creates a violation of both tactile and visual sensory expectations, the toy block is easily differentiated from the table.

It is important to emphasize how much the sensory and motor nervous systems are intertwined. Too often neuroscientists study these systems separately, but they are highly interconnected and work in concert. Although there are brain areas specialized for sensing such as auditory cortex and visual cortex, and there are areas of the brain devoted to action such as the motor cortex, most of the cortex is associational and cannot be called exclusively sensory or motor systems. These associational cortical areas are highly interconnected and the delineation between perception and action becomes blurred (Fuster, 2004). The parietal cortex receives multimodal sensory inputs and is important for planning movements. By multimodal we mean that the brain area receives more than one sense: auditory, olfactory, taste, visual, touch, or vestibular. The frontal cortex also receives multimodal inputs and is important for decisions, control of actions, and action selection. These multimodal association areas have direct influence on what we perceive and how we move.

2.4. Degeneracy

Degeneracy is the ability of elements that are structurally different to perform the same function or yield the same output (Edelman and Gally, 2001). To be fault tolerant and flexible a system's architecture should be designed such that different subsystems have different functional processes and there is an overlap of functionality between subsystems. In this design, if any subsystem fails the overall system can still function. This is different from redundancy, in which an identical system copy is kept in case there is a system failure (e.g., redundant array of independent disk [RAID] computer memory systems). Degeneracy appears throughout biology, from low-level processes such as the genetic code and protein folding to system-level processes such as behavioral repertoires and language. For example, there are four nucleotide bases in DNA (thymine, cytosine, adenine, and guanine). It takes three bases to encode an amino acid, which is the building block of proteins. This means that there are potentially 4^3 or sixty-four possible combinations, but only twenty amino acids make up the proteins found in the human body. In many cases, different triplets encode the same amino acid. Therefore, the genetic code is considered degenerate. As a result, the genetic code is fault tolerant to mutations. The heterogeneity of neuron types within and between brain regions, as well as between organisms is another example of degeneracy. For instance, the nervous system has numerous cell types which can be distinguished by their anatomy, connectivity or firing behavior (Ascoli et al., 2007; Wheeler et al., 2015). Furthermore, organisms like the nematode C. Elegans have neurons that don't fire action potentials (Sarma et al., 2018). Despite this variability, these neuronal elements communicate with sensors, actuators, and other brain regions that have similar properties and often operate in the same environment. At the other end of the biological spectrum is communication. We have an almost infinite number of ways to communicate the same message. The same message could be communicated through voice, text, email, Morse code, gesture, or facial expressions. Degeneracy and variability is not only important to demonstrate for biological realism, it also leads to robustness and fault tolerance when operating in noisy, dynamic environments.

Degeneracy at multiple levels was nicely demonstrated by the neurorobot Darwin X, which was used to demonstrate spatial and episodic memory (Krichmar et al., 2005; Fleischer et al., 2007). Darwin X solved a dry version of the Morris water maze and a place-learning version of a plus maze (see **Figure 2**). In the Morris water maze, a rat swims through murky water until it finds a platform hidden beneath the surface. After several days of exploration, the rat will swim directly to the platform from any starting location. If the hippocampus is damaged, the rat cannot learn the location of the platform. In the dry version, a reflective piece of paper was the proxy for the platform. It was the same color as the floor, so the robot could not see the platform but it could "feel" when it was on the platform by means of a downward-pointing light sensor.

Darwin X had an extensive model of the hippocampus formation and its surrounding cortical regions. Similar to a rodent, as the robot explored its environment, hippocampal place cells emerged. In the Darwin X experiments, combinations of place cells were used to plan routes to goals. The robot's behavior and neural activity were directly compared with rodent experiments. Like the rat, place cells could be used to predict not only the current location of the robot but also the location from which the robot came and the location to which it was heading. In the robot experiments, several levels of degeneracy emerged.

Degeneracy at the Neuronal Level

Because the neuroroboticists were able to track every neuron in Darwin's simulated nervous system, they were able to trace the neuronal activity that led to hippocampal place activity. Although hippocampal place activity was similar on different

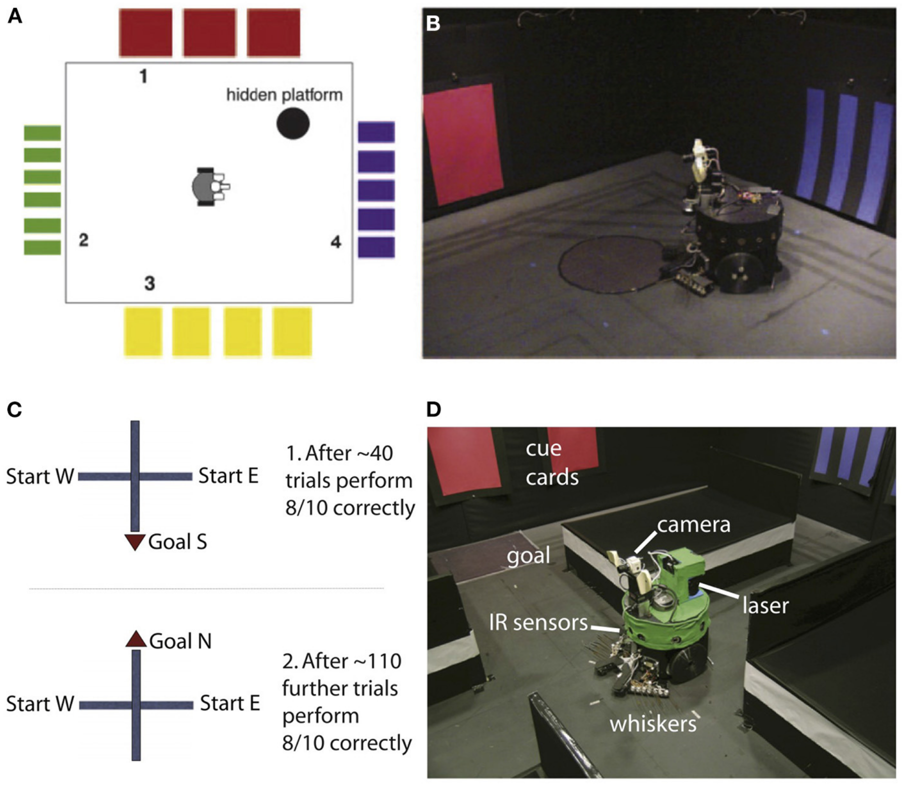


FIGURE 2 | Darwin X experiments for spatial and episodic memory. **(A,B)** Experimental setup for a dry version of the Morris water maze. Adapted from Krichmar et al. (2005). **(C,D)** Experimental setup for for place learning with the plus maze. Adapted from Fleischer et al. (2007).

trials when the robot passed through the same location on the same heading, the neuronal activity leading to that neuron's place activity on a given trial differed dramatically. That is, different neural activation patterns led to the same hippocampal place cell outcome.

Degeneracy at the Systems Level

Darwin X received sensory input from its camera (vision), whiskers (somatosensory), compass (head direction), and laser range finder (depth/distance). Darwin X's spatial memory was multimodal and degenerate. Even when one or more of its sensory modalities were lesioned, Darwin X's behavior and place cell activity remained stable. Different sensory pathways led to the same outcome of knowing where Darwin was located.

Degeneracy at the Behavioral Level

In the Morris maze task, nine different Darwin X subjects, which consisted of the identical robots with slightly different nervous systems due to variations in initial synaptic connection probabilities, solved the same spatial navigation task in unique ways. Some subjects bounced off the "red" wall to the hidden platform, some bounced off the "blue" wall, and others went directly toward the platform location. The proficiency of each subject differed as well. Some were better learners than others. However, despite their idiosyncrasies they all shared the same outcome of solving this task.

2.5. Multitasking and Event-Driven Processing

Cognitive scientists tend to study the brain in a serialized fashion by focusing on one subsystem at a time, be it a type of memory or a specific perceptual effect. But intelligence emerges from many processes operating in parallel and driven by events. We (humans and other organisms) are multitaskers, and to multitask we must do things in parallel. The brain is the ultimate event-driven, parallel computer. There is no overarching clock as in computer architectures. Rather the brain responds to events when they happen. Therefore, neurorobots should have a multitask design that responds to multiple, asynchronous events in a timely manner.

Neurons throughout the brain are responding simultaneously to multiple events. Although the different parts of the brain may be acting somewhat independently, they are highly interactive. The sensory system is telling the motor system what it senses, and the motor system is telling the sensory system what its last action was. This brain analogy can be extended to the whole organism, in which control is parallel, asynchronous, and spatiotemporally matched with the real world.

A classic way of studying cognitive science was a serial process of "sense, think, and act," which guided many artificial intelligence robot designs. In response to this, Rodney Brooks and Ron Arkin developed behavior-based robots that responded asynchronously to events (Brooks, 1991; Arkin, 1998). There are parallels between the subsumption architecture's layered hierarchical design and the nervous system. Neuroscientist Larry Swanson proposed a basic plan for the nervous system that somewhat follows this design; a four-component functional systems model with a motor system controlling behavior and visceral functions (i.e., internal organs and bodily functions), whose output is a function of activity in sensory, cognitive, and behavioral state systems, with feedback throughout (Swanson, 2007). It should be noted that in this view the central nervous system is only one component. Areas that regulate basic behaviors and internal monitoring are subcortical.

Correlates of the subsumption architecture can be observed in the brain's structure. Prescott et al. (1999) suggested that the hierarchical, layered subsumption architecture could describe the neural mechanisms of defense for the rat. For example, the lower levels were reactive and included withdrawal, startling, and freezing. The higher levels subsumed the lower levels by suppressing behavior or predicting outcomes through conditioning. Sensory input provides stimuli that can trigger behavioral responses. Another example involves self-monitoring systems in the nervous system (Chiba and Krichmar, 2020). **Figure 3** shows an overview of this architecture with brain systems on the left and the corresponding robot control systems on the right. There is low-level control for sensor processing and motor reflexes. The autonomic nervous system may subsume these lower levels to maintain homeostasis or adapt set points. Higher levels can set states or context that may shape responses.

Multitasking and event-driven processing is prevalent in current technology due in part to the ubiquity of real-time and embedded systems. Most modern computing devices, including smartphones, desktop computers, onboard automotive computers and entertainment systems, have parallel processes to handle asynchronous events. Neuromorphic computing architectures developed by researchers and major chip companies such as IBM and Intel are asynchronous and highly parallel, and they are composed of many computing units that act like neurons (Merolla et al., 2014; Davies et al., 2018). This architectural design also follows the *Efficiency through Cheap Design* principle described in Section 2.2. Neuromorphic hardware architectures use orders of magnitude less power than conventional computing by not relying on a synchronous clock and using spiking elements (Krichmar et al., 2019). A neuron uses most of its energy when it fires an action potential and when an action potential is processed at the synapse. Because neurons do not fire often (in the typical range of 10–100 Hz), the nervous system is in low-power mode between spikes. This idea was not lost on most neuromorphic hardware designers. Furthermore, communication bandwidth is reduced because information is sent only when there is a spike event.

3. ADAPTIVE BEHAVIOR - LEARNING AND MEMORY

Adaptation requires learning and remembering what was learned so that it can be applied in the future. Motivation is a key driver of learning. Motivators take many forms, which are called value systems. Another key aspect of adaptive behavior is the ability to predict future events. This requires building up a memory of expectations and the ability to adjust when expectations do not meet the current situation.

3.1. Learning and Memory

Unlike artificial systems, our brains allow us to learn quickly, incrementally, and continuously. With just a few presentations of something new, we can learn to recognize an object or situation or even learn a new skill. When we learn something new, we do not forget what we have learned previously. Moreover, we can take what we learn from one situation and apply it to another. On the other hand, artificial learning systems struggle under these situations, suffer from catastrophic forgetting of previous learning when something new is learned, and have difficulties generalizing learning from one task to another.

A brain region important for learning and memory is the hippocampus. The hippocampus is necessary to learn new memories and to consolidate those new experiences into long-term memories that can last a lifetime. The hippocampus can rapidly learn new autobiographical and semantic information, sometimes in the first experience (i.e., one-shot learning). Over time, this information becomes consolidated in the rest of the brain. Having a rapid learning system that can interact with a slower long-term storage area, which has been called complementary learning systems (McClelland et al., 1995; Kumaran et al., 2016), is thought to be the means by which our brains overcome catastrophic forgetting (i.e., forgetting previously learned information when learning new information). This aligns with another memory model, known as the hippocampal indexing theory (Teyler and DiScenna, 1986), which states that memories in the form of neocortical activation patterns are stored as indices in the hippocampus that are later used to aid recall. Although this may be an oversimplification, the notion that the hippocampus and medial temporal lobe integrates multimodal information from the neocortex makes sense and is backed by experimental evidence.

Our memories have context, and this contextual information can help us generalize when we encounter novel yet similar situations. In the literature this is called a schema, which is the memory of a set of items or actions bound together by a common context (van Kesteren et al., 2012). For example, if you are in a restaurant, you expect to see tables, chairs, a menu, waiters, and so forth. If you go to a new restaurant, that common context information can be used to rapidly consolidate the novel information into the restaurant schema. This requires mental representations that are flexible enough to learn tasks in new contexts and yet stable enough to retrieve and maintain tasks in old contexts (Carpenter and Grossberg, 1987). Tse et al. (2007) demonstrated this by training rats on different schemas, which were collections of associations between different foods and their locations in an enclosure. They found that the rats were able to learn new information quickly if it fit within a familiar schema. Additionally, the rats were able to learn new schemas without forgetting previous ones. The hippocampus was necessary for learning schemas and any new information matching a schema.

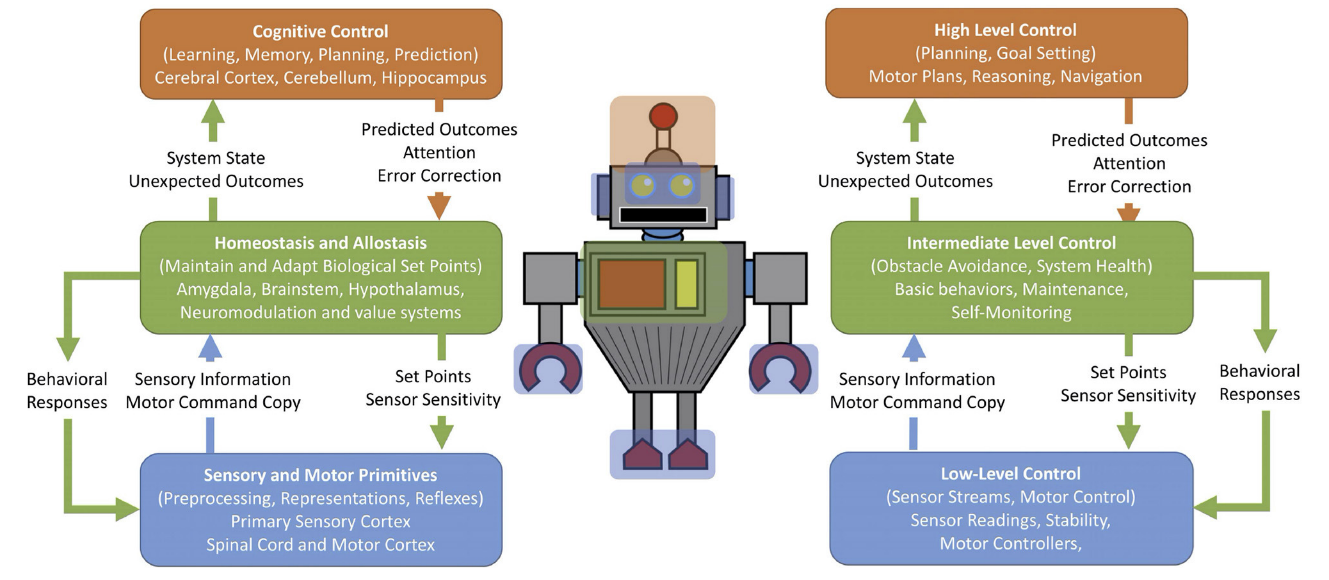

Cognitive Control
(Learning, Memory, Planning, Prediction)
Cerebral Cortex, Cerebellum, Hippocampus
System State
Unexpected Outcomes
Predicted Outcomes
Attention
Error Correction
Homeostasis and Allostasis
(Maintain and Adapt Biological Set Points)
Amygdala, Brainstem, Hypothalamus,
Neuromodulation and value systems
Behavioral Responses
Sensory Information
Motor Command Copy
Set Points
Sensor Sensitivity
Sensory and Motor Primitives
(Preprocessing, Representations, Reflexes)
Primary Sensory Cortex
Spinal Cord and Motor Cortex
High Level Control
(Planning, Goal Setting)
Motor Plans, Reasoning, Navigation
System State
Unexpected Outcomes
Predicted Outcomes
Attention
Error Correction
Intermediate Level Control
(Obstacle Avoidance, System Health)
Basic behaviors, Maintenance,
Self-Monitoring
Sensory Information
Motor Command Copy
Set Points
Sensor Sensitivity
Behavioral Responses
Low-Level Control
(Sensor Streams, Motor Control)
Sensor Readings, Stability,
Motor Controllers,

FIGURE 3 | Schematic for self-monitoring systems in biology and in engineering. On the left are terms and regions derived from neuroscience. On the right are terms adapted from autonomous robots but could be applied to many embedded systems. Blue: low-level sensory processing and motor control. Green: homeostasis, maintenance, and monitoring. Orange: high-level planning, adapting, and goal-driven behavior. Adapted from Chiba and Krichmar (2020).

A subsequent study showed increased plasticity in the medial prefrontal cortex (mPFC) when information was consistent with a familiar schema (Tse et al., 2011). However, the hippocampus was not necessary to recall these memories, even after a short time period (e.g., 48 h). This challenged the idea of complementary learning systems because new information could rapidly be consolidated in cortical memory under these conditions.

The Tse et al. (2007) schema experiment was replicated with a neural network model based on interactions between the hippocampus and the mPFC (Hwu and Krichmar, 2020), and later tested on a robot required to create and utilize a schema (Hwu et al., 2020). A contextual pattern of objects and locations projected to the mPFC, in which each individual neuron encoded a different schema. The ventral hippocampus (vHPC) and dorsal hippocampus (dHPC) created triplet indices of schema, object, and location. The vHPC and dHPC drove activity that eventually activated a place cell through a winner-take-all process in the action network. This action neuron caused the robot to go to that location in search of the cued object. Contrastive Hebbian learning (CHL) was used to make the association between schema, object, and location. CHL utilizes oscillatory epochs, in which the duration depends on the familiarity and novelty of information, to learn associations. This assimilation of new information has similarities to adaptive resonance theory or ART (Grossberg, 2013). The model contained neuromodulators to encode novelty and familiarity. For example, if an object is novel and the context is unfamiliar, a new schema must be learned. However, if an object is novel and the context is familiar, the object can be added to an existing schema.

The schema model was embedded on the Human Support Robot (HSR) from Toyota (Yamamoto et al., 2018) and given the task of finding and retrieving objects in a classroom and a break room. In a trial, the robot was prompted to retrieve an object, which required prior knowledge of the schema to which the object belonged and the location of the object. This caused the robot to navigate toward a location, recognize the object, grasp the object, and then return the object to its starting location.

In the first experiment, the HSR was placed in a room with typical classroom items (e.g., apple, bottle, computer mouse, book). **Figure 4A** shows the performance of the HSR retrieving classroom items. With each trial the number of correct places recalled increased and the time to retrieve an item decreased. After training and testing in the classroom, an original item was replaced by a novel item (Exp 1b in **Figure 4A**). Although the object was novel, the HSR knew it belonged in the classroom and was able to quickly consolidate this new information into the existing classroom schema.

In the second experiment, the HSR was placed in a room with typical break room items (e.g., apple, cup, banana, microwave oven). After training and testing, the classroom schema was tested again to see if the robot was able to maintain performance of prior tasks. As with the classroom, **Figure 4B** shows that the robot was able to learn this new break room schema without forgetting objects' locations in the classroom (CR in **Figure 4B**).

In the third experiment, the HSR was tested to see whether schemas could help with the retrieval of items that it was never explicitly trained to retrieve. If the HSR was cued with a book, it searched for the book on the desk in the classroom because books are likely to be found in a classroom schema (see **Figure 5A**). If the HSR was cued by showing it a banana, the HSR searched for the banana in the break room first rather than in the classroom (see **Figure 5B**).

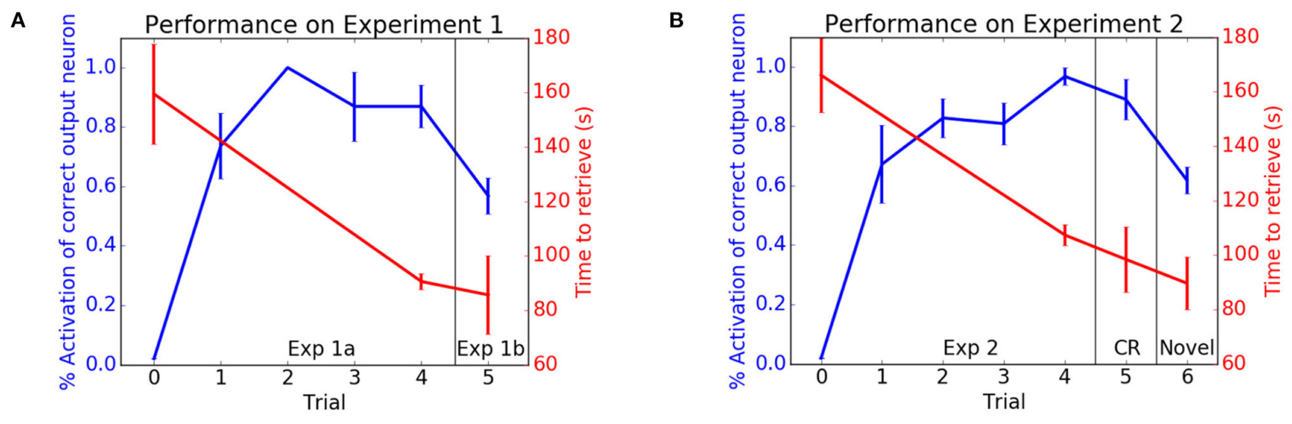


FIGURE 4 | Performance on neurobotic schema experiments. The blue lines show the activation of the correct location neuron for a cued object. The red lines show the retrieval times. **(A)** Experiment 1. Performance in the classroom schema (Exp 1a) and retrieval of novel object (Exp 1b). **(B)** Experiment 2. Performance in the breakroom schema. CR denotes performance when returned to classroom. Novel denotes retrieval of a novel object. Adapted from Hwu et al. (2020).

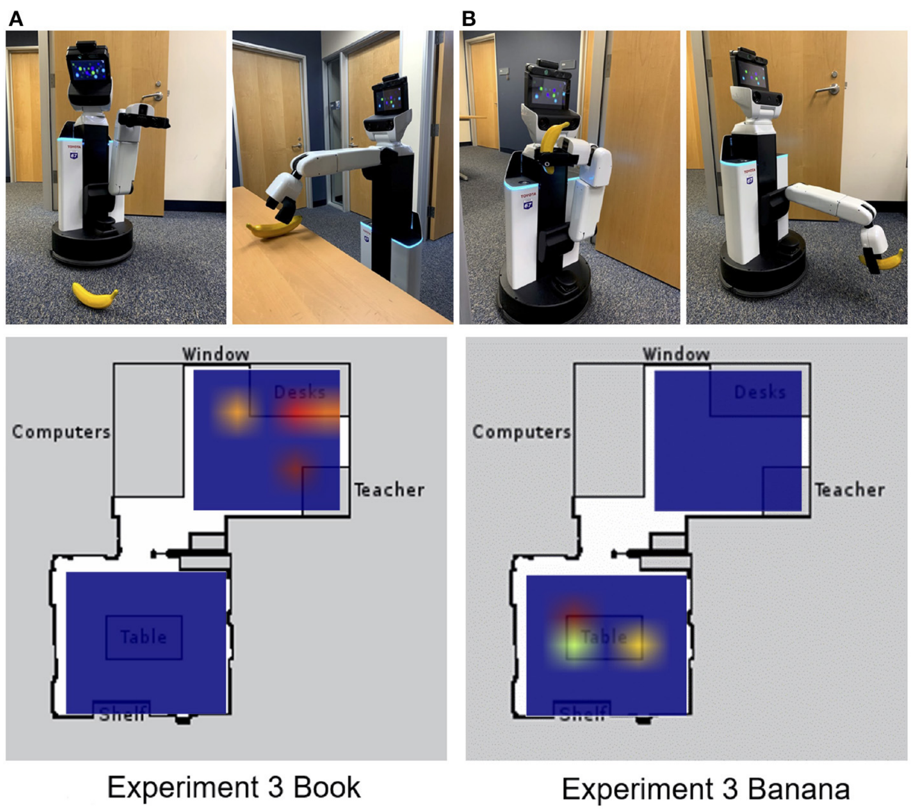


FIGURE 5 | Cuing the robot on objects it had not retrieved before. Top: The robot was shown a banana. The robot then went to the break room to pick up the banana and navigated to the drop-off location to deposit it. Bottom: Heat map of action layer during experiment 3. **(A)** Cued to retrieve a book. **(B)** Cued to retrieve a banana. Adapted from Hwu et al. (2020).

The neurorobotic schema experiments showed how context is tied to spatial representations via hippocampal interaction with the mPFC. Moreover, it demonstrates how ideas from memory models in the brain may improve robotic applications and issues in artificial intelligence, such as catastrophic forgetting and lifelong learning. A robot that has contextual memory could have applications for assistive technologies.

3.2. Value Systems

Robots should be equipped with a value system that constitutes a basic assumption of what is good and bad for its health and well-being. A value system facilitates the capacity of a biological brain to increase the likelihood of neural responses to an external phenomenon (Merrick, 2017). The combined effects of perception, experience, reasoning, and decision making contribute to the development of values in animals. Value can also be thought of as a measure of the effort one is willing to expend to obtain reward or to avoid punishment.

In addition to rewards and punishment, intrinsic motivation can be considered as a value system (Oudeyer and Kaplan, 2007). This can take the form of seeking novelty, fun, play or curiosity, that is, obtaining value for its own sake rather than to satisfy some need. For example, Oudeyer and colleagues created a robotic playground where a Sony AIBO dog with a motivation to explore and learn new things learned to manipulate objects (Oudeyer et al., 2007). The robot first spent time in situations that were easy to learn and then shifted its attention progressively to more difficult situations, avoiding situations in which nothing could be learned.

Neuromodulators are thought to act as the brain's value systems (Krichmar, 2008). Neuromodulators are chemicals that signal important environmental or internal events. They cause organisms to adapt their behavior through long-lasting signals to broad areas of the nervous system. Neuromodulators in the brain influence synaptic change (i.e., learning and memory) to satisfy global needs according to value.

To shape behavior, cognitive robots should have an innate value system to tell the robot that something was of value and trigger the appropriate reflexive behavior. From this experience the agent can learn which stimuli were predictive of that value and try to maximize the acquisition of good value while minimizing the acquisition of bad value. Many of these value-based robots employ models of the dopaminergic neuromodulatory system to shape behavior (Sporns and Alexander, 2002; Cox and Krichmar, 2009; Fiore et al., 2014; Chou et al., 2015).

Besides the dopaminergic reward system, there are multiple neuromodulators signal different value types (Doya, 2002; Krichmar, 2008). The serotonergic system is involved in harm aversion or impulsiveness (Miyazaki et al., 2018). The noradrenergic system signals oddball or unexpected events (Yu and Dayan, 2005). The cholinergic system is thought to increase attention to important features and at the same time to decrease the allocation of attention to distractors (Yu and Dayan, 2005). Acetylcholine and noradrenaline could be thought to signal intrinsic value by allocating attention and triggering learning (Avery and Krichmar, 2017). These systems interact with each other through direct and indirect pathways, and they all respond strongly to novelty by sending broad signals to large areas of the brain to cause a change in network dynamics resulting in decisive action.

Introducing saliency into the environment can lead to attentional signaling. For example, the robot CARL was designed to test a computational framework for applying neuromodulatory systems to the control of autonomous robots (Cox and Krichmar, 2009). The framework was based on the following premises (Krichmar, 2008): (1) the common effect of the neuromodulatory systems is to drive an organism to be decisive when environmental conditions call for such actions and allow the organism to be more exploratory when there are no pressing events; and (2) the main difference between neuromodulatory systems is the environmental stimuli that activate them. In the experiment, two out of four objects were salient, and CARL learned the appropriate action for each (see **Figure 6**). Unexpectedly, a strong attentional bias toward salient objects, along with ignoring the irrelevant objects, emerged through its experience in the real world. The selective attention could be observed both in CARL's behavior and in CARL's simulated brain. These neurorobotic experiments showed how phasic neuromodulation could rapidly focus attention on important objects in the environment by increasing the signal-to-noise ratio (SNR) of neuronal responses. The model further suggested that phasic neuromodulation amplifies sensory input and increases competition in the neural network by gating inhibition.

The neuromodulatory system also regulates attention allocation and response to unexpected events. Using the Toyota Human Support Robot (Yamamoto et al., 2018), the influence of the cholinergic (ACh) system and noradrenergic (NE) systems on goal-directed perception was studied in an action-based attention task (Zou et al., 2020). In this experiment, a robot was required to attend to goal-related objects (the ACh system) and adjust to the change of goals in an uncertain domain (the NE system). Four different actions (i.e., eat, work-on-computer, read, and say-hi) were possible in the experiment and each of them was associated with different images of objects. For example, the goal action “eat" might result in attention to objects such as apple or banana, whereas the action “say-hi" should increase attention to a person. During the experiment, the goal action changed periodically and the robot needed to select the action and object that it thought the user wanted on the basis of prior experience. The ACh system tracked the expected uncertainty about which goal was valid, and the NE system signaled unexpected uncertainty when goals suddenly changed (Yu and Dayan, 2005). High ACh activity levels allocated attention to different goals. Phasic NE responses caused a rapid shift in attention and a resetting of prior goal beliefs. The model demonstrated how neuromodulatory systems can facilitate rapid adaptation to change in uncertain environments. The goal-directed perception was realized through the allocation of the robot's attention to the desired action/object pair. **Figure 7** shows the robot deciding which object to bring to the user. The bottom of **Figure 7** shows views from the robot's camera as it correctly guesses that the user's goal is to eat. Its top-down attention system finds an appropriate object, which is an apple in this case.

One problem that remains unsolved in neurorobotics is that these artificial value systems are dissociated from the agent's

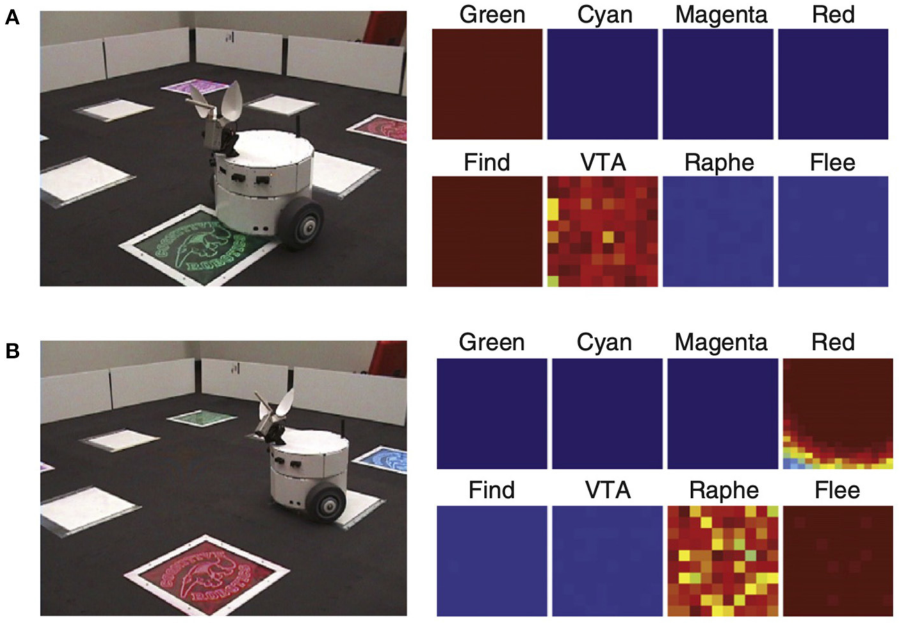


FIGURE 6 | CARL robot in colored panel task. The panels could flash any of six different colors. One color, green, signaled positive value. Another color, red, signaled negative value. The remaining colors were neutral. Positive and negative signals were transmitted from the panel to a receiver on the bottom of CARL. **(A)** CARL during an approach or find response. The panels on the right show strong neuronal activity in its simulated green visual area, the dopaminergic system (VTA), and the find motor neurons. **(B)** CARL during a withdrawal or flee response. The panels on the right show strong neuronal activity in its simulated red visual area, the serotonergic system (raphe), and the flee motor neurons. Adapted from Cox and Krichmar (2009).

body. Real pain, hunger, thirst, and fatigue drive a true value system. Without this connection to body-dependent drives, an artificial value system does not signal the immediacy of the agent's need and lacks to some degree the ability to shape the agent's behavior. It would be interesting to tie something like the robot's battery level to its hunger value. With faster-charging batteries or better solar cells this might be possible. An interesting step in this direction is the work on self-monitoring systems that can recognize drops in their performance, adapt their behavior, and recover. For example, Cully et al. (2015) developed a novel method for adapting gaits on a hexapod robot. In a sense, the robot controller had a memory of potential gaits. If one or more of the robot's legs were damaged, the robot would detect the damage, "imagine" different ways of moving, and then choose the new gait that it thought would work best under the new circumstances. In this way, the robot knew something was wrong and was able to adapt its behavior quickly without intervention.

3.3. Prediction

Predicting outcomes and planning for the future is a hallmark of cognitive behavior. Much of the cortex is devoted to predicting what we will sense or the outcome of a movement or what series of actions will lead to big payoffs. Thus, a neurorobot should strive to have these predictive capabilities. Through prediction and active inference, agents anticipate future sensory input based on prior experience. This minimizes free energy by predicting future outcomes so that they minimize the expenditures required to deal with unanticipated events (Friston, 2010). The idea of minimizing free energy has close ties to many existing brain theories, such as Bayesian brain, Predictive Coding, cell assemblies, Infomax, and the theory of neuronal group selection (Edelman, 1993; Rao and Ballard, 1999; Friston, 2010). In the theory of neuronal group selection (Edelman, 1993), plasticity is modulated by value. Value systems control which neuronal groups are selected and which actions lead to evolutionary fitness; that is, they predict outcomes that lead to positive value and avoid negative value. In this sense predicting value is inversely proportional to surprise.

Prediction is crucial for fitness in a complex world and a fundamental computation in cortical systems (George and Hawkins, 2009; Clark, 2013; Richert et al., 2016). It requires the construction and maintenance of an internal model. In a similar fashion, model-based reinforcement learning builds an internal model made up of the likelihood and expected value for transitions between states (Solway and Botvinick, 2012).

Prediction has been useful in developing robot controllers. For example, in a humanoid robot experiment it was shown that having a predictive model helped the robot make appropriate reactive and proactive arm gestures (Murata et al., 2014). In the proactive mode the robot's actions were generated on the basis of top-down intentions to achieve intended goals. In the reactive mode the robot's actions were generated by bottom-up sensory inputs in unpredictable situations. In another robot experiment the combination of model-based and model-free reinforcement learning was used in a sorting task (Renaudo et al., 2015). The robot had to push cubes on a conveyor belt. The model-based

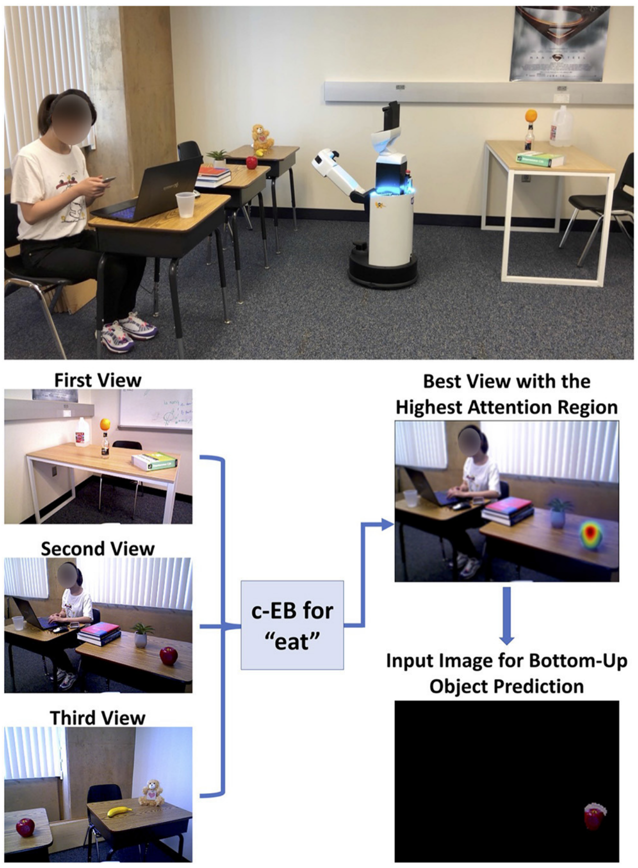

First View
Second View
Third View
c-EB for "eat"
Best View with the Highest Attention Region
Input Image for Bottom-Up Object Prediction

FIGURE 7 | Toyota Human Support Robot implementation for the goal-driven perception model, including the top-down attentional search process for a guessed action "eat" based on three different real indoor views, to select the highest attention region for bottom-up object prediction. Adapted from Zou et al. (2020).

system improved performance by maintaining a plan from one decision to the next. However, the experiments suggested that the model-free system scales better under certain conditions and may be better in the face of uncertainty.

Jun Tani's group has developed several predictive robot controllers using recurrent neural networks (Tani, 2016; Ahmadi and Tani, 2019; Chame et al., 2020). For example, they trained a hierarchy of continuous time recurrent neural networks (CTRNN) to learn different movements. Learning was achieved via backpropagation through time (BPTT), a means to apply an error signal to a sequence of neural activities. A teacher guided a humanoid Sony QRIO robot through different behavioral tasks. The CTRNNs received visual information and proprioceptive joint angles from the humanoid robot. Important to the learning were the different timescales of the CTRNNs. Slower higher-level CTRNNs sent predictions to the faster lower-level CTRNN units. Prediction errors from the lower levels were propagated to the higher levels for adjustments. Movements that appeared repeatedly were segmented into behavioral primitives. These primitives were represented in fast context dynamics in a form that was generalized across object locations. On the other hand, the slow context units appeared to be more abstract in nature, representing sequences of primitives in a way that was independent of the object location. Tani (2016) speculated that this prediction multiple timescale hierarchy had similarities to the cortex. Fast responding motor primitives can be found in the primary motor cortex, and the slower prefrontal cortex sends top-down predictions to the primary motor cortex. Similarly, the primary visual cortex sends sensory information and prediction errors to the slower parietal cortex, which sends top-down predictions for the primary visual cortex. In this group's recent work, they show the potential for prototyping robotics agents, modeled after active inference from the free energy principle theory (Friston, 2010), for human-robot interaction and socially assistive robotics (Chame et al., 2020).

4. BEHAVIORAL TRADE-OFFS - CONTEXTUAL DECISION-MAKING

Biological organisms need to consider many trade-offs to survive. These trade-offs regulate basic needs such as whether to forage for food, which might expose oneself to predators, or hide in one's home, which is safer but does not provide sustenance. These trade-offs can be cognitive as in introverted or extroverted behavior. Interestingly, many of these trade-offs are regulated by chemicals in our brain and body, such as neuromodulators or hormones. These modulatory areas monitor and regulate environmental events. They send broad signals to the brain that can dramatically change behaviors, moods, decisions, etc. The brain can control these modulatory and hormonal systems by setting a context or making an adjustment when there are prediction errors (Chiba and Krichmar, 2020).

We discuss the neuroscience behind the trade-offs and neurorobots that incorporate these trade-offs. We consider these behavioral trade-offs to be neurorobotic design principles. By applying them to neurorobots, we may realize behavior that is more interesting and more realistic.

4.1. Reward vs. Punishment

Dopaminergic neurons have phasic responses that match quite well with a reward prediction error signal used to shape behavior (Schultz et al., 1997). What about punishment? One model suggested that tonic serotonin tracked the average punishment rate and that tonic dopamine tracked the average reward rate (Daw et al., 2002). They speculated that a phasic serotonin signal might report an ongoing prediction error for future punishment. It has been suggested that the serotonergic and dopaminergic systems activate in opposition for goal-directed actions (Boureau and Dayan, 2011). This trade-off between reward and punishment can be quite nuanced when invigoration of activity can lead to rewards and punishment can lead to inaction.

4.2. Invigorated vs. Withdrawn

The dopamine and serotonin systems also regulate a trade-off between invigorated novelty seeking and withdrawn risk-averse behavior. It has been suggested that serotonin modulates the desire to withdraw from risk, which can take place in social interactions or in foraging for food (Tops et al., 2009). One can imagine that too much withdrawal from society could lead to symptoms of depression.

Consider the open-field test that is used to measure anxiety in rodents (Fonio et al., 2009). Usually, when a mouse is placed in an unfamiliar open area it will first stay near the borders of the environment in which it might be concealed. The mouse may hide in a nest area if available. After some time the mouse decides the environment is safe and becomes curious. The mouse will then proceed to explore the environment by moving more and investigating the middle of the area. Serotonin levels can alter this behavior. For example, Heisler et al. (1998) showed that mice with increased serotonin spent less time in the center of the open-field arena. In contrast, cocaine, which increases the level of dopamine, increased locomotive activity and the exploration of novel objects (Carey et al., 2008).

A neurorobot experiment took this trade-off into consideration by modeling the interactions between the serotonergic and dopaminergic systems (Krichmar, 2013). **Figure 8** shows the experimental setup and behaviors. A neural network controlled the behavior of an autonomous robot and tested in the open-field paradigm. When simulated serotonin levels were high, sensory events led to withdrawn anxious behavior such as wall following and finding its nest (i.e., the robot's charging station). When simulated dopamine levels were high, sensory events led to curious behavior such as locomotion to the middle of the enclosure or exploring a novel object.

The robot responded appropriately to sensory events in its environment. Novel objects resulted in its exploring the environment, and stressful events caused the robot to seek safety. When the environment was unfamiliar, serotonergic activity dominated, resulting in anxious behavior such as WallFollow and FindHome actions. However, once the robot had become more familiar with its environment (approximately 60 s into the trial) DA levels were higher and there was more curious or exploratory behavior. At approximately 120 s into the trial, there was an unexpected light event due to flashing the lights on and off, which resulted in a phasic 5-HT response and a longer tonic increase in 5-HT. This caused the robot to respond with withdrawn or anxious behavior until approximately 210 s into the trial when a pair of object events triggered exploration of the center of the environment. Specifically, tonic levels of 5-HT had decayed, and the object events caused an increase in DA levels triggering a change in behavioral state (see **Figure 9A**). **Figure 9B** shows the proportion of curious behavior (OpenField and ExploreObject) and anxious behavior (FindHome and WallFollow) for five experimental trials. Each bar was the average proportion of time spent in either curious (green bars) or anxious (red bars) behaviors. The error bar denoted the standard error. **Figure 9C** shows the behavior time-locked to the light event. The light event, which occurred at approximately the halfway point in the trial, was introduced to cause a stress response. After the light event, the neurorobot's behavior rapidly switched to anxious behavior until roughly 60 s later when it became curious again. Variation occurred due to different times of the light event and random variations in other sensory events.

Similar to the rodent experiments, changing the serotonin and dopamine levels affected the robot's behavior. Increasing the tonic serotonin levels in the model caused the robot to respond to a stressful event such as a bright light to stay near the walls or its charging station indefinitely. Increasing the tonic DA levels resulted in more curiosity and risk taking. In effect, it took more risks by venturing into the middle of the environment during

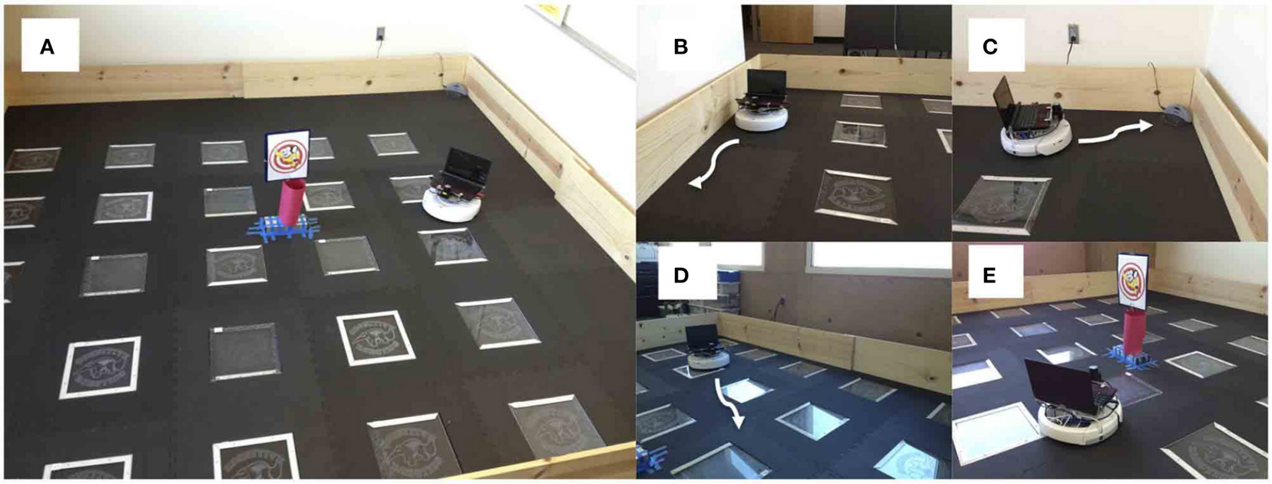


FIGURE 8 | Neurorobotic experimental setup for invigorated and withdrawn behavior. Experiments were run on an iRobot Create. **(A)** Robot arena. The picture in the middle was a novel object for the robot to explore. **(B)** Wall follow behavior. **(C)** Find home behavior. Finding the robot's docking station. **(D)** Open field behavior. **(E)** Explore object behavior. Adapted from Krichmar (2013).

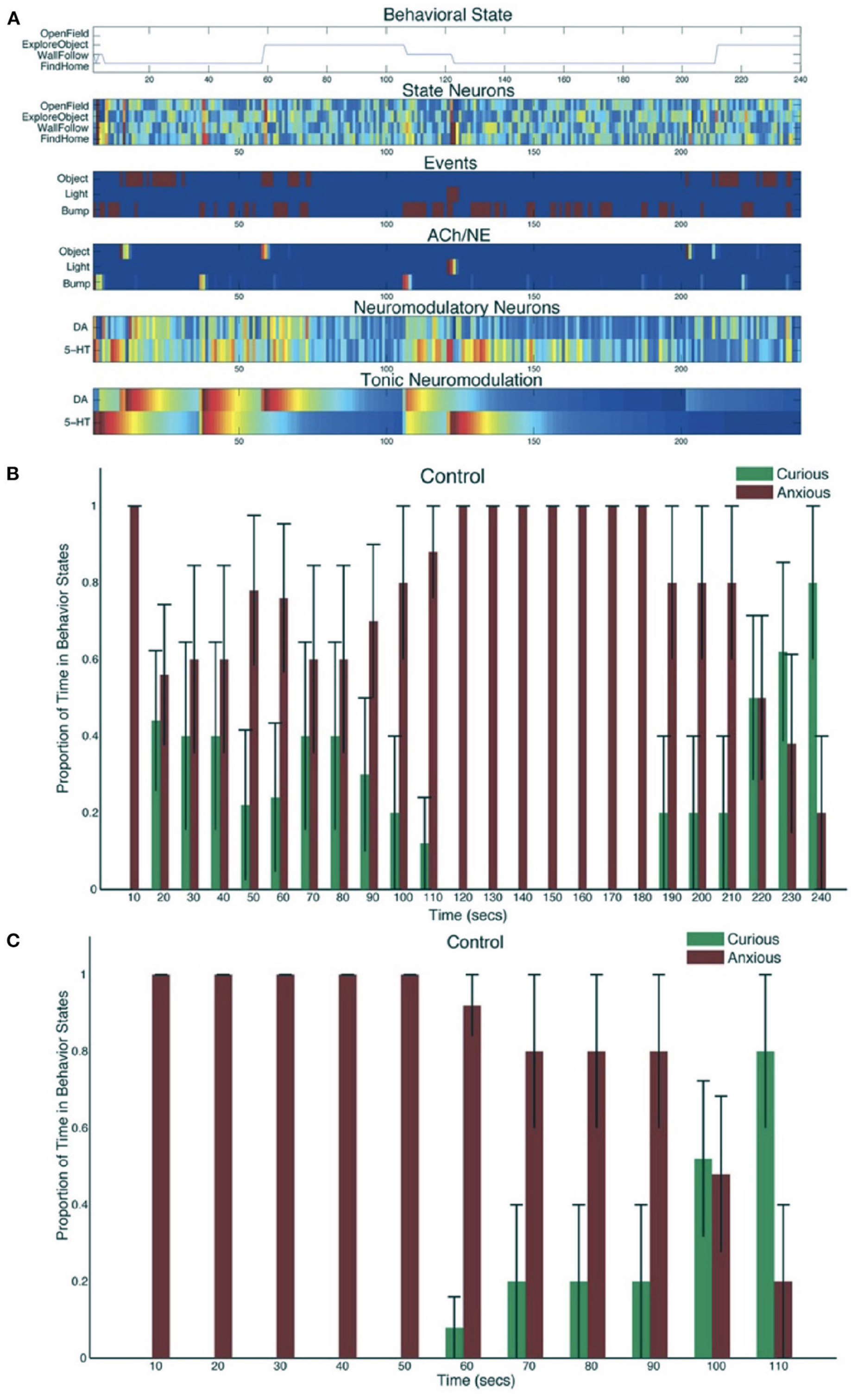


FIGURE 9 | Neurorobot results for invigorated and withdrawn behavior. **(A)** A representative example of behavioral and neural responses in a single trial. The x-axis shows time progression of the trial in seconds. The subplot for "Behavioral State" denotes the state of the robot across time. The subplots for "State Neurons," "Events," "ACh/NE," and "Neuromodulatory Neurons" show neural activity over the trial with pseudocolors ranging from dark blue (no activity) to bright red (maximal activity). The subplot for "Tonic Neuromodulation" denotes the level of tonic activation contributing to DA and 5-HT neurons. **(B)** Proportion of time in Curious (ExploreObject and OpenField) and Anxious (FindHome and WallFollow) behavior, averaged over 5 trials. Error bars represent standard error. **(C)** Similar to B except the behaviors were time-locked to the Light event. Adapted from Krichmar (2013).

or right after the stressful light event. Despite the simplicity of the neural network model, the robot's behavior looked quite natural and similar to that of a mouse in the same situation. Since the neuroroboticist was able to precisely control the neuromodulation activity, the experiment could shed light on neurological issues such as anxiety, depression, and obsessive compulsive disorders.

4.3. Expected Uncertainty vs. Unexpected Uncertainty

The world is full of uncertainty with which we must cope in our daily lives. Sometimes the uncertainty is expected, forcing us to increase our concentration on a task. Other times the uncertainty is unexpected, forcing us to divert our attention. How we deal with these types of uncertainty can be thought of as a behavioral trade-off. Once again, neuromodulators influence this trade-off of how we apply our attention. Yu and Dayan (2005) suggested that cholinergic neuromodulation tracks expected uncertainty (i.e., the known unreliability in the environment), and noradrenergic neuromodulation tracks unexpected uncertainty (i.e., observations that violate prior expectations) The basal forebrain, where cholinergic neurons reside, encodes the uncertainty of prior information and this can modulate attention to different features. The locus coeruleus, where noradrenergic neurons reside, is involved in cognitive shifts in response to novelty. When there are strong violations of expectations, locus coeruleus activity may induce a "network reset" that causes a reconfiguration of neuronal activity that clears the way to adapt to these changes (Bouret and Sara, 2005). In modeling and in experimental work, it has been shown that the cholinergic system mediates uncertainty seeking (Naude et al., 2016; Belkaid and Krichmar, 2020). Uncertainty seeking is especially advantageous in situations when reward sources are uncertain. The trade-off between expected and unexpected uncertainty can also be observed in how we apply our attention (Avery et al., 2012). Top-down attention or goal-driven attention, which ramps up our attention to look for something, is related to expected uncertainty. Bottom-up or stimulus-driven attention occurs when a surprise or unexpected uncertainty diverts our attention.

The Toyota HSR neurorobot experiment discussed in Section 3.2 and shown in **Figure 7** explored this trade-off between expected and unexpected uncertainty by modeling the cholinergic and noradrenergic system ability to regulate attention (Zou et al., 2020). Because the user's goals could be uncertain, simulated cholinergic neurons tracked how likely the user would be to choose any of these goals (i.e., expected uncertainty). When the user interacting with the robot changed their goals (i.e., unexpected uncertainty), the noradrenergic system in the model responded by resetting prior beliefs and rapidly adapting to the new goal. **Figure 10** shows how the robot correctly guessed the user's goals, which then drove attention to the object associated with the goal (e.g., eat leads to attention to an apple or orange). Note how quickly the noradrenergic (NE) system responded to goal changes, which led to the cholinergic system (ACh) increasing attention to objects related to the new goal. In this way, the robot was able to monitor a trade-off between the known and unknown uncertainties in the world to rapidly respond to the user's changing needs. Similar to the example in the schema experiments described in Section 3.1, such goal-driven attention could provide benefits for assistive robot technologies.

4.4. Exploration vs. Exploitation

During decision making or information gathering, there exists a trade-off between exploration and exploitation in which it is sometimes best to *explore* new options and other times it is best to *exploit* opportunities that have paid off in the past. A framework was presented in which neuromodulation controlled the exploration/exploitation trade-off (Aston-Jones and Cohen, 2005). When neuromodulators have tonic activity, the animal's behavior is exploratory and somewhat arbitrary. However, when the neuromodulator has a burst of phasic activity, the animal is decisive and exploits the best potential outcome at the given time. The CARL robot described in Section 3.2 (see **Figure 6**) incorporated tonic and phasic neuromodulation (Cox and Krichmar, 2009). When there was tonic neuromodulation, the robot randomly explored its environment by looking at different colored panels. If one of the colors became salient due to a reward or punishment signal, the robot's neuromodulatory systems responded with a phasic burst of activity. This phasic neuromodulation caused a rapid exploitative response to investigate the colored panel. It also triggered learning to approach positive-value objects and avoid negative-value objects.

4.5. Foraging vs. Defending

Hormones are chemical messengers in the body that can affect the brain and other organs. They regulate a number of bodily functions such as body temperature, thirst, hunger, and sleep. Like neuromodulators, hormones can be triggered by environmental events and can broadly change neural activity. For example, the hormone orexin regulates hunger levels. This can lead to a behavioral trade-off in which animals foraging for food are less willing to defend a territory (Padilla et al., 2016). Foraging for food may cause an animal to leave its nest exposed to predators. However, defending one's territory requires energy expenditure, which if prolonged requires food for replenishment.

Hormones can be modeled and embodied in robots to explore interesting naturalistic tradeoffs (Canamero, 1997). For example, Cañamero's group modeled hormones that tracked a robot's health (Lones et al., 2018); one hormone was related to the battery level, and another hormone monitored the robot's internal temperature, which was related to how much the robot moved and the climate of its environment. The robot's tasks were to maintain health and gather food resources, which might require aggressive action. This neurorobotics study demonstrated how maintaining health requires behavioral trade-offs. Searching for food increased the robot's internal temperature and reduced the robot's battery level. Being aggressive to obtain food also reduced battery levels. However, not searching for food would lead to starvation. Modeling the secretion and decay of hormones allowed the robot to maintain a comfortable energy and internal temperature level and at times led to an aggressive behavior of pushing objects away to get at food resources. However,

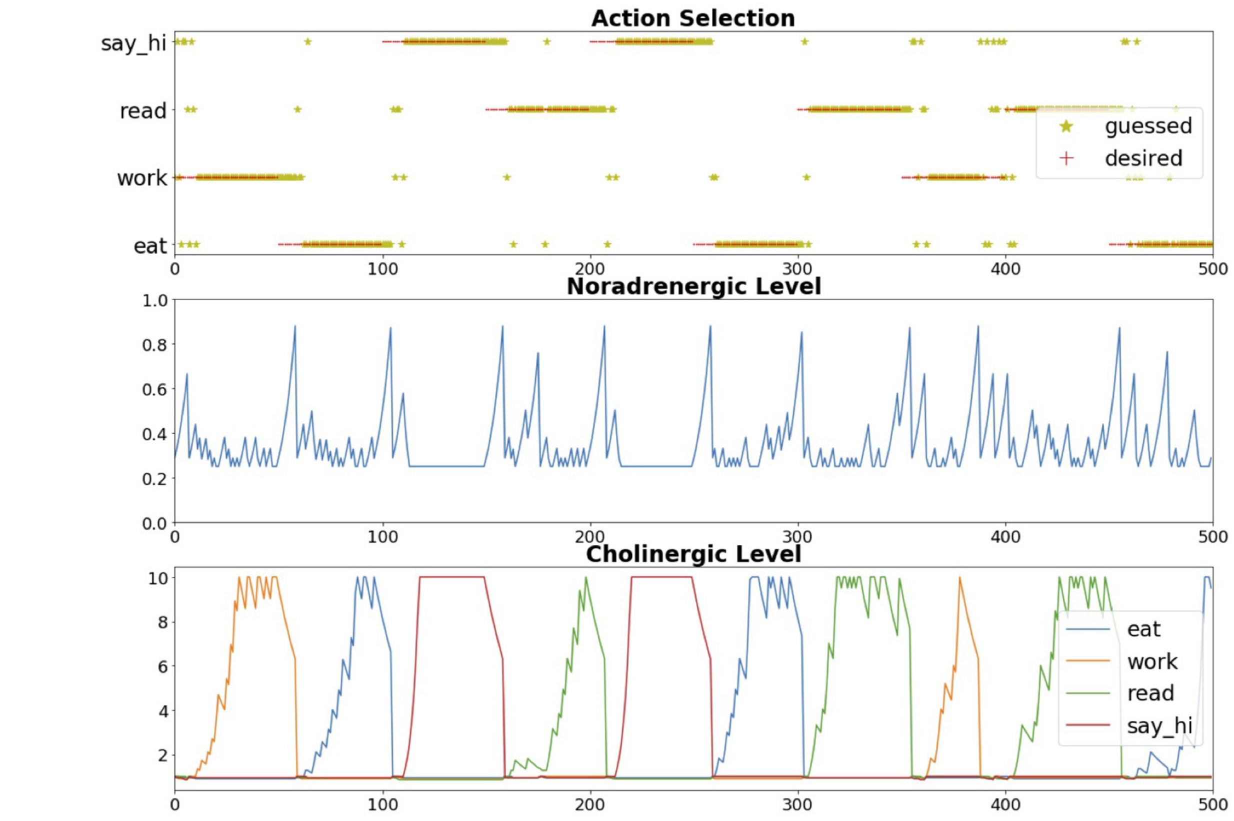


FIGURE 10 | Expected and unexpected uncertainty neuromodulation in neurorobot experiment. Top chart. Robot's response to guess the user's goal. Center chart. Noradrenergic (NE) neuron activity level. Bottom chart. Cholinergic (ACh) neuron activity level. There was an ACh neuron for each potential goal. Adapted from Zou et al. (2020).

as the environments became more complex an epigenetic system, which monitored and controlled the hormones, became necessary for the robot's comfort level to be maintained satisfactorily. Their epigenetic system acted in a similar way to the hypothalamus, a subcortical brain region that regulates many of our bodily functions. Their experiments showed that an epigenetic mechanism significantly and consistently improved the robot's adaptability and might provide a useful general mechanism for adaptation in autonomous robots.

4.6. Stress vs. Calm

In his book Why Zebras Don't Get Ulcers, Robert Sapolsky describes how a zebra, which is calmly grazing, responds when it encounters a lion (Sapolsky, 2004). The zebra quickly runs away from this stressful situation. Once clear of danger, the zebra is calm again. This fight-or-flight response is mediated by the stress hormones known as glucocorticoids, which increase blood flow and awareness. However, this stress response does come at the expense of regulating long-term health and short-term memory (Chiba and Krichmar, 2020). Unlike zebras, people sometimes remain in a constant state of stress due to elevated glucocorticoids, which can cause damage to the hippocampus and memory.

Although there has been little work to date on neurorobots that regulate their stress level, downregulation of behavior could be useful for autonomous systems far from power sources, which might have a stress-like response to carry out a mission and then switch to a low-power calm mode after the mission has been accomplished. In an interesting paradigm that explores the stress vs. calm trade-off, experiments have shown that rats appear to be capable of empathy and prosocial behavior (Ben-Ami Bartal et al., 2011). In one study a rat was trapped in a cage and clearly stressed. Another rat observing this behavior became stressed, too. The observing rat, feeling bad for its trapped friend, found a lever that opened the cage and released the trapped rat. This study suggested that rats can feel another's pain (i.e., feel empathy) and are willing to act on the other's behalf (i.e., can be prosocial).

In a robotic variation of the empathy experiment, a rat was trapped in a cage interacted with two different robots, one of which was helpful and opened the cage and the other of which was uncooperative and ignored the trapped rat (Quinn et al., 2018). Interestingly, the rat remembered who its robot friends were. When the helpful robot was trapped, the rat freed that robot but did not free the robot that was uncooperative. This could have implications for rescue robotics. A robot that can identify and relieve stress or anxiety could have applications for robotic caretakers or for disaster relief.

4.7. Social vs. Solitary

Hormones can regulate a trade-off between social bonding and independence. Estrogen, progesterone, oxytocin, and prolactin can influence a number of neural systems to ensure maternal nurturing, bonding, and protection of young (Rilling and Young, 2014). In particular, oxytocin has been shown to regulate social and paternal bonding (Young and Wang, 2004). An interesting example of oxytocin's effect on bonding has been observed in voles. Whereas, prairie voles are polygamous and the male does not assist in the nurturing of young pups, meadow voles are monogamous and both parents participate in the pup rearing. Interestingly, meadow voles have more oxytocin receptors than prairie voles. Furthermore, inhibiting oxytocin prevents pair bonding in meadow voles (Young and Wang, 2004). However, social bonding requires devoting energy to another, possibly at the expense of one's own health. Therefore, it could be argued that there should be a balance between social and solitary behavior.

Neurorobot experimenters have investigated the balance between social and solitary behavior. By simulating social hormones, Cañamero's group investigated attachment bonds between a robot and a "caregiver" (Canamero et al., 2006; Hiolle et al., 2009, 2012). Although they did not explicitly model oxytocin, their system did simulate the type of bonding observed between a parent and a child. The robot found a good balance between asking the caregiver for help and learning on its own. Too much interaction with a caregiver led to stress and rejection by the robot. Not enough interaction with a caregiver resulted in isolation. As with humans, a proper balance is important for learning and development.

5. DISCUSSION

In this article, we discussed a number of principles to consider when designing neurorobots and experiments using robots to test brain theories. These principles are strongly inspired by Pfeifer and Bongard's design principles for intelligent agents. We build upon these design principles by grounding them in neurobiology and by adding principles based on neuroscience research. We highlight the importance in neurorobotics for designing systems that are reactive, adaptive, predictive, able to manage behavioral tradeoffs, and capable of learning from experience.

As summarized in **Figure 11**, the principles fall into three broad categories: 1) *Embodiment and reactions.* These are reactive, reflexive and rapid responses. They are often carried out without involving the central nervous system. Rather they emerge through the body's interaction with the environment or are handled by the peripheral nervous system and reflex arcs involving the spinal cord. They can have short-term adaptive properties and lead to behavioral repertoires. 2) *Adaptive behavior.* Biological organisms have the ability to learn continually over the lifetime of the organism. A major property of the brain is its plasticity. In particular, hippocampal interactions with the cortex lead to long-term contextual memory. Neuromodulatory systems can signal value, which shapes learning and triggers adaptive behavior. Another hallmark of the brain is its ability to predict outcomes. This requires the construction, maintenance and updating of memory systems. 3) *Behavioral trade-offs.* To survive in a dynamic world, organisms must make decisions based on their needs and environmental context. Oftentimes, these needs are a trade-off between opposing motivations (e.g, taking a risk for a reward vs. playing it safe to avoid punishment). These trade-offs can lead to interesting behavior. Many of these trade-offs are regulated by sub-cortical neuromodulator and hormone levels.

5.1. Importance of Low-Level Processes and Model Organisms

Although the examples in this article focused primarily on vertebrates, the principles could be applied to modeling other organisms. Studies of the insect visual system have led to elegant, efficient solutions for robot navigation that could be deployed on neuromorphic hardware (Galluppi et al., 2014; Schoepe et al., 2021). The emphasis on vertebrates and especially the mammalian brain has been data-driven in part. There are numerous studies of rodents, non-human primates and humans that have provided modelers with anatomical, behavioral, and neurophysiological data to make their simulations more biologically accurate. However, the complexity of these organisms and their brains makes holistic modeling difficult. A promising avenue may be to study organisms that have less complex brains and behaviors. Recent work on model organisms such as drosophila and the nematode C. Elegans are providing rich data sets for neuroboticists (Sarma et al., 2018; Scheffer et al., 2020). The OpenWorm project (Sarma et al., 2018) provides the biological data and the simulation tools, including a robotics sub-project, to create interesting neurorobots that follow many of the design principles introduced here.

There is a tendency in neurorobotics, which is a subarea of cognitive robotics (Cangelosi and Asada, 2022), to model cognitive functions such as attention, decision-making, planning, etc. Although modeling human cognition may be an ultimate goal for the field, many of the design principles introduced here concentrate on low-level processes such as how the body shapes behavior, motivations, homeostatic control of body functions to name a few. As we emphasized in this paper and in Chiba and Krichmar (2020), many of these processes are driven by sub-cortical brain regions and neurochemicals. Their interaction with the environment and bodily functions lead to interesting behavior that could be described as cognitive. Early examples of neurorobots and behavior-based robots demonstrated that intelligent behavior could arise from interactions between the robot and the environment without complex nervous systems or control systems (Braitenberg, 1986; Brooks, 1991; Holland, 2003). More work needs to be done to first build a foundation of these low-level processes, upon which higher-order cognitive processes can be added. Moreover, each cognitive model should carefully choose an appropriate level of abstraction and state assumptions about the lower level processes that support the model.

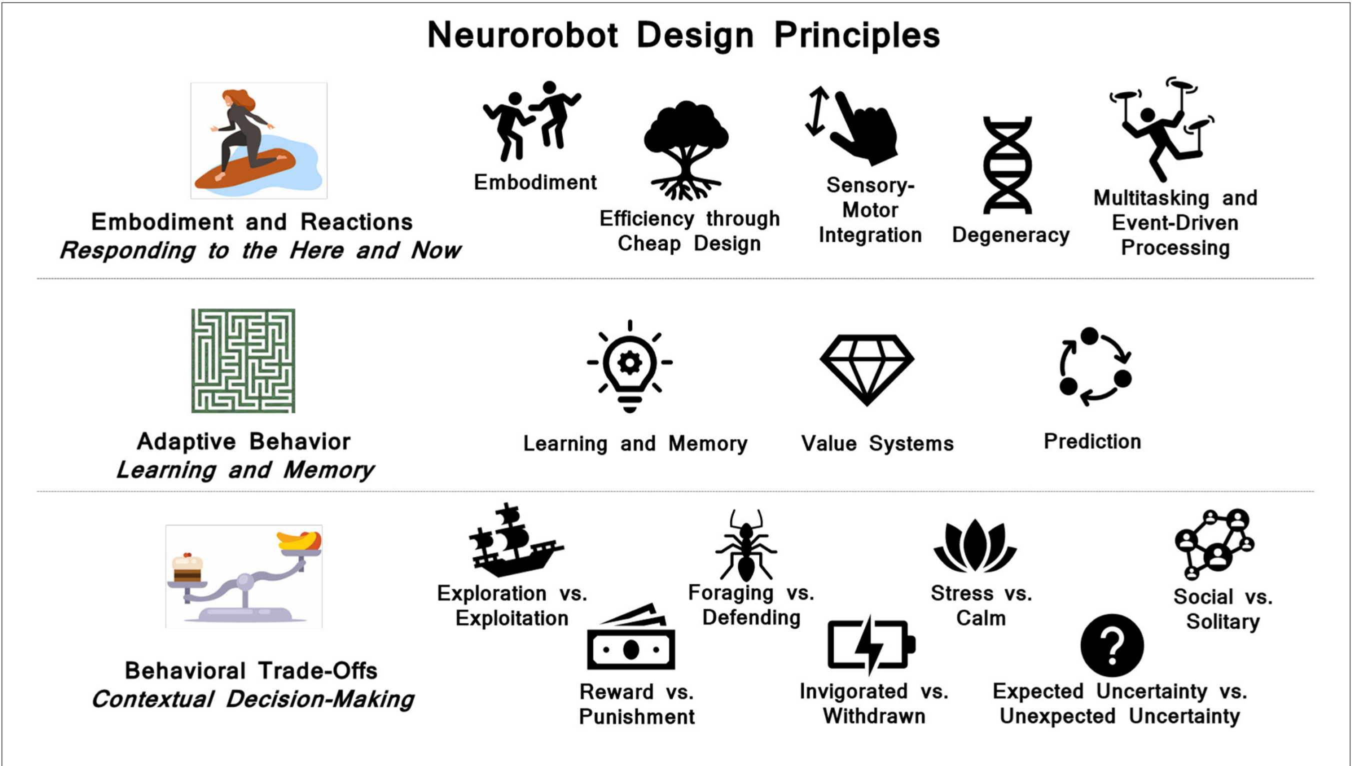

Neurorobot Design Principles
Embodiment and Reactions
Responding to the Here and Now
Embodiment
Efficiency through Cheap Design
Sensory-Motor Integration
Degeneracy
Multitasking and Event-Driven Processing
Adaptive Behavior
Learning and Memory
Learning and Memory
Value Systems
Prediction
Behavioral Trade-Offs
Contextual Decision-Making
Exploration vs. Exploitation
Reward vs. Punishment
Foraging vs. Defending
Invigorated vs. Withdrawn
Stress vs. Calm
Expected Uncertainty vs. Unexpected Uncertainty
Social vs. Solitary

FIGURE 11 | Summary of neurorobot design principles.

5.2. Next Steps and Future Directions

The principles of neurorobotics introduced here can address major challenges facing artificial intelligence and robotics research. In general, neurorobotics explores these challenges in embodied settings, providing a fresh perspective that extends beyond simulations and algorithms performed on computers. Neurorobotics incorporates features of the brain that may begin to address lifelong continual learning, efficient computing, operating on scarce knowledge, and human-computer interaction.

We expect that new neuroscience discoveries will further inform neurorobots, and vice versa. Progress in learning and memory may lead to applications capable of continual learning. Advances in our understanding of multimodal sensory systems may be incorporated into neurorobotics that not only classify but also understand the meaning of what they are perceiving. Given the recent achievements of artificial intelligence, hybrid systems combining machine learning and deep learning with neurorobotic design principles could lead to interesting applications in autonomous driving, assistive robots, and manufacturing.

In the intermediate term, we believe that neurobiological concepts in learning and memory, navigation, decision making, social behavior, and more, will have found their way into practical applications. Progress in neuromorphic computing and algorithms will lead to applications that can run at the edge with little human intervention. This may lead to advances in search and rescue robots and in robots capable of autonomously exploring unknown environments such as the deep sea or extraterrestrial planets.

In the long-term we hope that neurorobotics will achieve more general intelligence rather than being designed for specific tasks. In fact the delineation between conventional robotics and neurorobotics may be blurred, with all robots possessing some neurobiologically inspired aspects. With the rapid advances in computing and other technologies, it is hard to predict far into the future. However, we do believe that neurorobotics and cognitive machines, in some form, will seamlessly be a part of our everyday lives.

6. CONCLUSION

In closing, we believe that neurorobots that follow many of the design principles discussed in this article will have more interesting, naturalistic behavior. This not only allows the robot to be a better model for understanding the complex behaviors observed in biology, it also could lead to better robots that show more intelligence and that are more natural in their interactions with other agents.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

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

FUNDING

This work was supported by NSF award IIS-1813785 and by the United States Air Force Award FA9550-19-1-0306.

ACKNOWLEDGMENTS

The authors would like to thank the present and former members of the CARL lab at UC Irvine for many fruitful discussions and inspiring robot experiments.

REFERENCES

Ahmadi, A., and Tani, J. (2019). A novel predictive-coding-inspired variational rnn model for online prediction and recognition. *Neural Comput*. 31, 2025–2074. doi: 10.1162/neco_a_01228

Arkin, R. (1998). *Behavior-Based Robotics*. Cambridge, MA: MIT Press.

Ascoli, G. A., Donohue, D. E., and Halavi, M. (2007). Neuromorpho.org: a central resource for neuronal morphologies. *J. Neurosci*. 27, 9247–9251. doi: 10.1523/JNEUROSCI.2055-07.2007

Aston-Jones, G., and Cohen, J. D. (2005). An integrative theory of locus coeruleus-norepinephrine function: adaptive gain and optimal performance. *Annu. Rev. Neurosci*. 28, 403–450. doi: 10.1146/annurev.neuro.28.061604.135709

Avery, M. C., and Krichmar, J. L. (2017). Neuromodulatory systems and their interactions: a review of models, theories, and experiments. *Front. Neural Circ*. 11, 108. doi: 10.3389/fncir.2017.00108

Avery, M. C., Nitz, D. A., Chiba, A. A., and Krichmar, J. L. (2012). Simulation of cholinergic and noradrenergic modulation of behavior in uncertain environments. *Front. Comput. Neurosci* 6, 5. doi: 10.3389/fncom.2012.00005

Belkaid, M., and Krichmar, J. L. (2020). Modeling uncertainty-seeking behavior mediated by cholinergic influence on dopamine. *Neural Netw*. 125, 10–18. doi: 10.1016/j.neunet.2020.01.032

Ben-Ami Bartal, I., Decety, J., and Mason, P. (2011). Empathy and pro-social behavior in rats. *Science* 334, 1427–1430. doi: 10.1126/science.1210789

Beyeler, M., Rounds, E. L., Carlson, K. D., Dutt, N., and Krichmar, J. L. (2019). Neural correlates of sparse coding and dimensionality reduction. *PLoS Comput. Biol*. 15, e1006908. doi: 10.1371/journal.pcbi.1006908

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

Bouret, S., and Sara, S. J. (2005). Network reset: a simplified overarching theory of locus coeruleus noradrenaline function. *Trends Neurosci*. 28, 574–582. doi: 10.1016/j.tins.2005.09.002

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. Cambridge, MA: MIT Press. A Bradford Book.

Brooks, R. A. (1991). Intelligence without representation. *Artif. Intell*. 47, 139–159. doi: 10.1016/0004-3702(91)90053-M

Canamero, D. (1997). "Modeling motivations and emotions as a basis for intelligent behavior," in *Proceedings of the First International Conference on Autonomous Agents* (Marina del Rey, CA), 148–155.

Canamero, L., Blanchard, A. J., and Nadel, J. (2006). Attachment bonds for human-like robots. *Int. J. Humanoid Rob*. 3, 301–320. doi: 10.1142/S0219843606000771

Cangelosi, A., and Asada, M. (2022). *Cognitive Robotics*. Cambridge, MA: The MIT Press.

Carey, R. J., Damianopoulos, E. N., and Shanahan, A. B. (2008). Cocaine effects on behavioral responding to a novel object placed in a familiar environment. *Pharmacol. Biochem. Behav*. 88, 265–271. doi: 10.1016/j.pbb.2007.08.010

Carpenter, G. A., and Grossberg, S. (1987). Art 2: self-organization of stable category recognition codes for analog input patterns. *Appl. Opt*. 26, 4919–4930. doi: 10.1364/AO.26.004919

Chame, H. F., Ahmadi, A., and Tani, J. (2020). A hybrid human-neurorobotics approach to primary intersubjectivity via active inference. *Front. Psychol*. 11, 584869. doi: 10.3389/fpsyg.2020.584869

Chiba, A., and Krichmar, J. (2020). Neurobiologically inspired self-monitoring systems. *Proc. IEEE* 108, 976–986. doi: 10.1109/JPROC.2020.2979233

Chiel, H. J., and Beer, R. D. (1997). The brain has a body: adaptive behavior emerges from interactions of nervous system, body and environment. *Trends Neurosci*. 20, 553–557. doi: 10.1016/S0166-2236(97)01149-1

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

Clark, A. (2013). Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behav. Brain Sci*. 36, 181–204. doi: 10.1017/S0140525X12000477

Collins, S., Ruina, A., Tedrake, R., and Wisse, M. (2005). Efficient bipedal robots based on passive-dynamic walkers. *Science* 307, 1082–1085. doi: 10.1126/science.1107799

Cox, B. R., and Krichmar, J. L. (2009). Neuromodulation as a robot controller: a brain inspired design strategy for controlling autonomous robots. *IEEE Rob. Autom. Mag*. 16, 72–80. doi: 10.1109/MRA.2009.933628

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

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

Daw, N. D., Kakade, S., and Dayan, P. (2002). Opponent interactions between serotonin and dopamine. *Neural Netw*. 15, 603–616. doi: 10.1016/S0893-6080(02)00052-7

Doya, K. (2002). Metalearning and neuromodulation. *Neural Netw*. 15, 495–506. doi: 10.1016/S0893-6080(02)00044-8

Edelman, G. M. (1993). Neural darwinism: selection and reentrant signaling in higher brain function. *Neuron* 10, 115–125. doi: 10.1016/0896-6273(93)90304-A

Edelman, G. M., and Gally, J. A. (2001). Degeneracy and complexity in biological systems. *Proc. Natl. Acad. Sci. U.S.A*. 98, 13763–13768. doi: 10.1073/pnas.231499798

Fiore, V. G., Sperati, V., Mannella, F., Mirolli, M., Gurney, K., Friston, K., et al. (2014). Keep focussing: striatal dopamine multiple functions resolved in a single mechanism tested in a simulated humanoid robot. *Front. Psychol*. 5, 124. doi: 10.3389/fpsyg.2014.00124

Fitzpatrick, P., and Metta, G. (2003). Grounding vision through experimental manipulation. *Philos. Trans. A Math. Phys. Eng. Sci*. 361, 2165–2185. doi: 10.1098/rsta.2003.1251

Fleischer, J., Szatmary, B., Hutson, D., Moore, D., Snook, J., Edelman, G., et al. (2006). "A neurally controlled robot competes and cooperates with humans in segway soccer," in *IEEE International Conference on Robotics and Automation* (Orlando, FL: IEEE), 3673–3678.

Fleischer, J. G., Gally, J. A., Edelman, G. M., and Krichmar, J. L. (2007). Retrospective and prospective responses arising in a modeled hippocampus during maze navigation by a brain-based device. *Proc. Natl. Acad. Sci. U.S.A*. 104, 3556–3561. doi: 10.1073/pnas.0611571104

Fonio, E., Benjamini, Y., and Golani, I. (2009). Freedom of movement and the stability of its unfolding in free exploration of mice. *Proc. Natl. Acad. Sci. U.S.A*. 106, 21335–21340. doi: 10.1073/pnas.0812513106

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nat Rev Neurosci* 11, 127–138. doi: 10.1038/nrn2787

Fuster, J. M. (2004). Upper processing stages of the perception-action cycle. *Trends Cogn Sci.* 8, 143–145. doi: 10.1016/j.tics.2004.02.004

Gallagher, S. (2001). The practice of mind: theory, simulation or interaction? *J. Consciousness Stud*. 8, 83–108. Available online at: https://www.ingentaconnect.com/contentone/imp/jcs/2001/00000008/f0030005/1207#

Galluppi, F., Denk, C., Meiner, M. C., Stewart, T. C., Plana, L. A., Eliasmith, C., et al. (2014). "Event-based neural computing on an autonomous mobile platform," in *2014 IEEE International Conference on Robotics and Automation (ICRA)* (Hong Kong: IEEE), 2862–2867.

Gaussier, P., Banquet, J. P., Cuperlier, N., Quoy, M., Aubin, L., Jacob, P.-Y., et al. (2019). Merging information in the entorhinal cortex: what can we learn from robotics experiments and modeling? *J. Exp. Biol*. 222, 1–13. doi: 10.1242/jeb.186932

George, D., and Hawkins, J. (2009). Towards a mathematical theory of cortical micro-circuits. *PLoS Comput. Biol.* 5, e1000532. doi: 10.1371/journal.pcbi.1000532

Grossberg, S. (2013). Adaptive resonance theory: how a brain learns to consciously attend, learn, and recognize a changing world. *Neural Netw*. 37, 1–47. doi: 10.1016/j.neunet.2012.09.017

Heisler, L. K., Chu, H. M., Brennan, T. J., Danao, J. A., Bajwa, P., Parsons, L. H., et al. (1998). Elevated anxiety and antidepressant-like responses in serotonin 5-ht1a receptor mutant mice. *Proc. Natl. Acad. Sci. U.S.A*. 95, 15049–15054. doi: 10.1073/pnas.95.25.15049

Hickok, G., Houde, J., and Rong, F. (2011). Sensorimotor integration in speech processing: computational basis and neural organization. *Neuron* 69, 407–422. doi: 10.1016/j.neuron.2011.01.019

Hiolle, A., Bard, K. A., and Canamero, L. (2009). "Assessing human reactions to different robot attachment profiles," in *Ro-Man 2009: The 18th IEEE International Symposium on Robot and Human Interactive Communication, Vols. 1, 2* (Toyama), 824–829.

Hiolle, A., Canamero, L., Davila-Ross, M., and Bard, K. A. (2012). "Eliciting caregiving behavior in dyadic human-robot attachment-like interactions," in *ACM Transactions on Interactive Intelligent Systems, Vol. 2*, Article 3.

Holland, O. (2003). Exploration and high adventure: the legacy of grey walter. *Philos. Trans. A Math. Phys. Eng. Sci*. 361, 2085–2121. doi: 10.1098/rsta.2003.1260

Hwu, T., Kashyap, H. J., and Krichmar, J. L. (2020). "A neurobiological schema model for contextual awareness in robotics," in *2020 International Joint Conference on Neural Networks (IJCNN)* (Glasgow), 1–8.

Hwu, T., and Krichmar, J. L. (2020). A neural model of schemas and memory encoding. *Biol. Cybern*. 114, 169–186. doi: 10.1007/s00422-019-00808-7

Hwu, T. J., and Krichmar, J. L. (2022). *Neurorobotics: Connecting the Brain, Body, and Environment*. Cambridge, MA: The MIT Press.

Krichmar, J. L. (2008). The neuromodulatory system - a framework for survival and adaptive behavior in a challenging world. *Adapt. Behav*. 16, 385–399. doi: 10.1177/1059712308095775

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., Nitz, D. A., Gally, J. A., and Edelman, G. M. (2005). Characterizing functional hippocampal pathways in a brain-based device as it solves a spatial memory task. *Proc. Natl. Acad. Sci. U.S.A*. 102, 2111–2116. doi: 10.1073/pnas.0409792102

Krichmar, J. L., Severa, W., Khan, M. S., and Olds, J. L. (2019). Making bread: biomimetic strategies for artificial intelligence now and in the future. *Front. Neurosci*. 13, 666. doi: 10.3389/fnins.2019.00666

Kumaran, D., Hassabis, D., and McClelland, J. L. (2016). What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends Cogn. Sci*. 20, 512–534. doi: 10.1016/j.tics.2016.05.004

Laughlin, S. B., and Sejnowski, T. J. (2003). Communication in neuronal networks. *Science* 301, 1870–1874. doi: 10.1126/science.1089662

Lones, J., Lewis, M., and Cañamero, L. (2018). A hormone-driven epigenetic mechanism for adaptation in autonomous robots. *IEEE Trans. Cogn. Dev. Syst*. 10, 445–454. doi: 10.1109/TCDS.2017.2775620

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychol. Rev*. 102, 419–457. doi: 10.1037/0033-295X.102.3.419

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). Artificial brains. a million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673. doi: 10.1126/science.1254642

Merrick, K. (2017). Value systems for developmental cognitive robotics: a survey. *Cogn. Syst. Res*. 41, 38–55. doi: 10.1016/j.cogsys.2016.08.001

Milford, M., Jacobson, A., Chen, Z., and Wyeth, G. (2016). Ratslam: Using models of rodent hippocampus for robot navigation and beyond. *Rob. Res. ISRR* 114, 467–485. doi: 10.1007/978-3-319-28872-7_27

Miyazaki, K., Miyazaki, K. W., Yamanaka, A., Tokuda, T., Tanaka, K. F., and Doya, K. (2018). Reward probability and timing uncertainty alter the effect of dorsal raphe serotonin neurons on patience. *Nat. Commun*. 9, 2048. doi: 10.1038/s41467-018-04496-y

Murata, S., Arie, H., Ogata, T., Sugano, S., and Tani, J. (2014). Learning to generate proactive and reactive behavior using a dynamic neural network model with time-varying variance prediction mechanism. *Adv. Rob*. 28, 1189–1203. doi: 10.1080/01691864.2014.916628

Naude, J., Tolu, S., Dongelmans, M., Torquet, N., Valverde, S., Rodriguez, G., et al. (2016). Nicotinic receptors in the ventral tegmental area promote uncertainty-seeking. *Nat. Neurosci.* 19, 471–478. doi: 10.1038/nn.4223

Olshausen, B., and Field, D. (2004). Sparse coding of sensory inputs. *Curr. Opin. Neurobiol.* 14, 481–487. doi: 10.1016/j.conb.2004.07.007

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

Oudeyer, P. Y., and Kaplan, F. (2007). What is intrinsic motivation? a typology of computational approaches. *Front. Neurorob*. 1, 6. doi: 10.3389/neuro.12.006.2007

Padilla, S. L., Qiu, J., Soden, M. E., Sanz, E., Nestor, C. C., Barker, F. D., et al. (2016). Agouti-related peptide neural circuits mediate adaptive behaviors in the starved state. *Nat. Neurosci*. 19, 734–741. doi: 10.1038/nn.4274

Pfeifer, R., and Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. Cambridge, MA: The MIT Press.

Prescott, T. J., Redgrave, P., and Gurney, K. (1999). Layered control architectures in robots and vertebrates. *Adapt. Behav*. 7, 99–127. doi: 10.1177/105971239900700105

Quinn, L. K., Schuster, L. P., Aguilar-Rivera, M., Arnold, J., Ball, D., Gygi, E., et al. (2018). When rats rescue robots. *Anim. Behav. Cogn*. 5, 368–379. doi: 10.26451/abc.05.04.04.2018

Rao, R. P., and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci*. 2, 79–87. doi: 10.1038/4580

Renaudo, E., Girard, B., Chatila, R., and Khamassi, M. (2015). Respective advantages and disadvantages of model-based and model-free reinforcement learning in a robotics neuro-inspired cognitive architecture. *Procedia Comput. Sci*. 71(Suppl. C), 178–184. doi: 10.1016/j.procs.2015.12.194

Richert, M., Fisher, D., Piekniewski, F., Izhikevich, E. M., and Hylton, T. L. (2016). Fundamental principles of cortical computation: unsupervised learning with prediction, compression and feedback. *arXiv:1608.06277*. doi: 10.48550/arXiv.1608.06277

Rilling, J. K., and Young, L. J. (2014). The biology of mammalian parenting and its effect on offspring social development. *Science* 345, 771. doi: 10.1126/science.1252723

Sapolsky, R. M. (2004). *Why Zebras Don't Get Ulcers: The Acclaimed Guide to Stress, Stress-Related Disease and Coping, 3rd Edn*. New York, NY: Holt Paperbacks.

Sarma, G. P., Lee, C. W., Portegys, T., Ghayoomie, V., Jacobs, T., Alicea, B., et al. (2018). Openworm: overview and recent advances in integrative biological simulation of caenorhabditis elegans. *Philos. Trans. R Soc. Lond. B Biol. Sci*. 373, 1758. doi: 10.1098/rstb.2017.0382

Scheffer, L. K., Xu, C. S., Januszewski, M., Lu, Z., Takemura, S. Y., Hayworth, K. J., et al. (2020). A connectome and analysis of the adult drosophila central brain. *Elife* 9, e57443. doi: 10.7554/eLife.57443.sa2

Schoepe, T., Janotte, E., Milde, M., Bertrand, O., Egelhaaf, M., and Chicca, E. (2021). Finding the gap: neuromorphic motion vision in cluttered environments. *arXiv:2102.08417 [cs.NE]*. doi: 10.48550/arXiv.2102.08417

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science* 275, 1593–1599. doi: 10.1126/science.275.5306.1593

Shadmehr, R., and Krakauer, J. W. (2008). A computational neuroanatomy for motor control. *Exp. Brain Res*. 185, 359–381. doi: 10.1007/s00221-008-1280-5

Solway, A., and Botvinick, M. M. (2012). Goal-directed decision making as probabilistic inference: a computational framework and potential neural correlates. *Psychol. Rev*. 119, 120–154. doi: 10.1037/a0026435

Sporns, O. (2010). *Networks of the Brain*. Cambridge, MA: MIT Press.

Sporns, O., and Alexander, W. H. (2002). Neuromodulation and plasticity in an autonomous robot. *Neural Netw*. 15, 761–774. doi: 10.1016/S0893-6080(02)00062-X

Swanson, L. W. (2007). Quest for the basic plan of nervous system circuitry. *Brain Res. Rev*. 55, 356–372. doi: 10.1016/j.brainresrev.2006.12.006

Tani, J. (2016). *Exploring Robotic Minds: Actions, Symbols, and Consciousness as Self-Organizing Dynamic Phenomena*. New York, NY: Oxford University Press.

Teyler, T. J., and DiScenna, P. (1986). The hippocampal memory indexing theory. *Behav. Neurosci*. 100, 147–154. doi: 10.1037/0735-7044.100.2.147

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

Tse, D., Langston, R. F., Kakeyama, M., Bethus, I., Spooner, P. A., Wood, E. R., et al. (2007). Schemas and memory consolidation. *Science* 316, 76–82. doi: 10.1126/science.1135935

Tse, D., Takeuchi, T., Kakeyama, M., Kajii, Y., Okuno, H., Tohyama, C., et al. (2011). Schema-dependent gene activation and memory encoding in neocortex. *Science* 333, 891–895. doi: 10.1126/science.1205274

van Kesteren, M. T., Ruiter, D. J., Fernandez, G., and Henson, R. N. (2012). How schema and novelty augment memory formation. *Trends Neurosci*. 35, 211–219. doi: 10.1016/j.tins.2012.02.001

Wheeler, D. W., White, C. M., Rees, C. L., Komendantov, A. O., Hamilton, D. J., and Ascoli, G. A. (2015). Hippocampome.org: a knowledge base of neuron types in the rodent hippocampus. *Elife* 4, e09960. doi: 10.7554/eLife.09960

Yamamoto, T., Nishino, T., Kajima, H., Ohta, M., and Ikeda, K. (2018). "Human support robot (hsr)," in *Siggraph'18: Acm Siggraph 2018 Emerging Technologies* (Vancouver, BC).

Young, L. J., and Wang, Z. (2004). The neurobiology of pair bonding. *Nat. Neurosci*. 7, 1048–1054. doi: 10.1038/nn1327

Yu, A. J., and Dayan, P. (2005). Uncertainty, neuromodulation, and attention. *Neuron* 46, 681–692. doi: 10.1016/j.neuron.2005.04.026

Ziemke, T. (2003). "What's that thing called embodiment?" in *Proceedings of the Annual Meeting of the Cognitive Science Society, Vol. 25* (Boston, MA), 1305–1310.

Zou, X., Kolouri, S., Pilly, P. K., and Krichmar, J. L. (2020). Neuromodulated attention and goal-driven perception in uncertain domains. *Neural Netw*. 125, 56–69. doi: 10.1016/j.neunet.2020.01.031

Conflict of Interest: TH was employed by Riot Games.

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

frontiers | Frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 26 May 2022
doi: 10.3389/fnbot.2022.867019

A Spiking Neural Network Model of Rodent Head Direction Calibrated With Landmark Free Learning

*Rachael Stentiford, Thomas C. Knowles and Martin J. Pearson**

Bristol Robotics Laboratory, University of the West England Bristol, Bristol, United Kingdom

Maintaining a stable estimate of head direction requires both self-motion (idiothetic) information and environmental (allothetic) anchoring. In unfamiliar or dark environments idiothetic drive can maintain a rough estimate of heading but is subject to inaccuracy, visual information is required to stabilize the head direction estimate. When learning to associate visual scenes with head angle, animals do not have access to the 'ground truth' of their head direction, and must use egocentrically derived imprecise head direction estimates. We use both discriminative and generative methods of visual processing to learn these associations without extracting explicit landmarks from a natural visual scene, finding all are sufficiently capable at providing a corrective signal. Further, we present a spiking continuous attractor model of head direction (SNN), which when driven by idiothetic input is subject to drift. We show that head direction predictions made by the chosen model-free visual learning algorithms can correct for drift, even when trained on a small training set of estimated head angles self-generated by the SNN. We validate this model against experimental work by reproducing cue rotation experiments which demonstrate visual control of the head direction signal.

Keywords: spiking neural network, pyNEST, head direction, predictive coding, localization, continuous attractor

OPEN ACCESS

Edited by:
Leslie Samuel Smith,
University of Stirling, United Kingdom

Reviewed by:
James Courtney Knight,
University of Sussex, United Kingdom
Dan Turner-Evans,
University of California, Santa Cruz, United States

****Correspondence:***
Martin J. Pearson
martin.pearson@brl.ac.uk

Received: *31 January 2022*
Accepted: *19 April 2022*
Published: *26 May 2022*

Citation:
Stentiford R, Knowles TC and Pearson MJ (2022) A Spiking Neural Network Model of Rodent Head Direction Calibrated With Landmark Free Learning.
Front. Neurorobot. 16:867019.
doi: 10.3389/fnbot.2022.867019

1. INTRODUCTION

As we move through the world we see, touch, smell, taste and hear the environment around us. We also experience a sense of our own self-motion through our vestibular system, which enables us to keep balance and to maintain an internal estimate of our location and heading—pose—in the world. Any drift in pose estimate incurred through the integration of self-motion cues alone (as we walk with our eyes closed for example) is quickly corrected when we open our eyes and recognize familiar features in the environment. This approach to self-localization has been adopted in many fields of engineering that require an accurate and persistent pose estimate to operate effectively, such as mobile robots and augmented reality devices.

The relative contributions of self-motion (idiothetic) and external sensory cues (allothetic) to the firing properties of 'spatial cells' in rodents has been extensively investigated in neuroscience (see below for review). The integration of idiothetic cues provides the animal with a rapid and constant estimate of pose. This estimate not only aids navigation in the absence of allothetic cues, but is also a learning scaffold to associate pose with novel visual scenes. This second function has received little attention in prior models which often use the ground truth pose of a learning mobile agent to associate sensory view rather than the drift prone estimate provided from idiothetic cues. In mobile robotics this problem is addressed in the research field known as Simultaneous Localization And Mapping (SLAM) with myriad solutions proposed, each with their own advantages and limitations based on environmental, sensory and computational constraints. In this study we are interested in modeling how this problem has been solved in the mammalian brain.

The problem of accumulative error in idiothetic cue integration implies that head direction estimates from early explorations of a novel environment should be more reliable than later explorations. Therefore, earlier experiences of an environment should be used to learn associations between heading and visual scenes, to correct for drift as the animal later explores the same environment. From a learning perspective this puts constraints on the size and richness of training sets. This bi-directional learning problem is investigated here through a series of controlled experiments using a simulated mobile robot within a virtual environment. Our contribution is to model the head direction cell system using populations of spiking neurons, translating angular head velocity cues into a spike encoded representation of head direction. To anchor head directions to allothetic cues, we have trained three different model-free visual learning algorithms: Convolutional Neural Network (CNN), Variational Auto-Encoder (VAE) and Predictive Coding Network (PCN), to associate distal natural visual scenes with the spike based representation of head direction. We demonstrate that all three models are capable of correcting the drift in pose estimate from purely idiothetic cue integration even when trained on small self-generated training sets. This is evaluated further through cue conflict experiments to reveal similar characteristics of the model performance as recorded in rodents. The primary motivation for this work is toward a deeper understanding of the complete head direction system of the rodent through the integration of components modeled at different levels of abstraction; namely, spiking neural attractor network models, deep learning based generative and discriminative models, and simulated robotic embodiment. However, the integration of models at multiple levels of abstraction also provides a framework for how energy efficient neuromorphic hardware components (Krichmar et al., 2019) can be usefully integrated into mobile robotic applications in the near future. We contend that to fully exploit this biologically inspired computing paradigm requires continued biomimetic study of fundamental neuroscience as epitomized in the field of neurorobotics.

1.1. Rodent Head Direction Cell System

Neural correlates of position (O'Keefe, 1976; Hafting et al., 2005), environmental boundaries (Lever et al., 2009), heading (Taube et al., 1990), speed (Kropff et al., 2015) and numerous other spatial measures (see Grieves and Jeffery, 2017 for review) have been extensively studied in the rodent brain and remain an active topic in neuroscience research. Of these, head direction (HD) cells—cells which exhibit high firing rates only in small arcs of head angle—appear simplest, and have been a popular target for modeling. Head direction cells have also been identified in regions homologous to the rodent hippocampus in birds (Ben-Yishay et al., 2021), fish (Vinepinsky et al., 2020), and insects (Kim et al., 2017). Strikingly in Drosophila these cells are arranged as a ring in the Ellipsoid Body, and have properties of a continuous attractor.

Most models of head direction use a continuous attractor, where a sustained bump of activity centered on the current heading is formed and maintained through interactions between excitatory and inhibitory cells. Many rely on recurrent excitatory collaterals between cells in the Lateral Mammillary Nuclei (LMN; Zhang, 1996; Page and Jeffery, 2018), however anatomical data show no evidence of this type of connection (Boucheny et al., 2005). Although head direction cells have been found in many brain regions, including Anterior Thalamic Nuclei (ATN; Taube, 1995), Retrosplenial cortex (Cho and Sharp, 2001), Lateral Mammillary nuclei (LMN; Stackman and Taube, 1998) and Dorsal Tegmental Nucleus (DTN; Sharp et al., 2001; see Yoder et al., 2011 for review), generation of the head direction signal is thought to be in the reciprocal connection between LMN and DTN (Blair et al., 1999; Bassett and Taube, 2001). As the DTN sends mainly inhibitory connections to the LMN, attractor networks exploiting connections between two populations of cells appear more biologically plausible (Boucheny et al., 2005; Song and Wang, 2005).

1.2. Control of HD by Self-Motion Cues

Self-motion cues can be derived directly from the vestibular system but also from optic flow and motor efference copy. Disrupting vestibular input to head direction cells abolishes spatial firing characteristics and impacts behaviors which rely on heading (Yoder and Taube, 2009, 2014). Cells sensitive to Angular Head Velocity (AHV) have been recorded in several regions including the DTN (Bassett and Taube, 2001; Sharp et al., 2001). These cells are either sensitive to AHV in a single direction (clockwise or anticlockwise; asymmetric AHV cells) or the magnitude of AHV regardless of direction (symmetric AHV cells). Methods of moving the bump of activity on the ring attractor to follow head movement rely mainly on asymmetric AHV input. Bump movement is achieved either through imbalance between two populations of cells in the attractor network (Boucheny et al., 2005; Bicanski and Burgess, 2016), or via conjunctive cells which fire strongly as a function of both AHV and head direction (Sharp et al., 2001; McNaughton et al., 2006). However, using imprecise self-motion cues in the absence of vision results in drift in the preferred firing direction of head direction cells (Stackman et al., 2003).

1.3. Visual Control of HD

Although HD cells still show some directional sensitivity in the absence of visual cues or novel environments (Goodridge and Taube, 1995; Taube and Burton, 1995; Goodridge et al., 1998; Stackman et al., 2003), vision is clearly an important factor for stabilizing the head direction system. During development, head direction cells have much sharper tuning curves after eye opening (Tan et al., 2015), but may use other types of allothetic information, such as tactile exploration of corners of the environment with whiskers, to stabilize head direction before eye opening (Bassett et al., 2018). Even in unfamiliar environments, visual information helps to stabilize head direction, suggesting ongoing learning of visual landmarks (Yoder et al., 2011). In familiar environments, the preferred firing directions of head direction cells become entrained to visual features and will follow these cues over self-motion signals (Taube and Burton, 1995). When environmental cues are rotated, preferred firing directions of many cells also rotate through the same angle, resulting in "bilateral" preferred firing directions; Page and Jeffery (2018) suggest these bilateral cells may be useful for assessing the stability of environmental landmarks. Some head direction cells won't follow these big conflicts in cue location, suggesting multiple populations of head direction cells that are more or less strongly controlled by allothetic input (Dudchenko et al., 2019). This visual control of head direction begins at the LMN (Yoder et al., 2015), stabilizing the head direction signal at its origin. Both the postsubiculum (PoS) and retrosplenial cortex (RSC) are likely candidates for delivering this visual information to the LMN (Taube et al., 1990; Vann et al., 2009). Head directions cells (or compass neurons) have also been shown to follow visual information in Drosophila (Fisher et al., 2019). In this case visual inputs onto compass neurons are inhibitory, and plasticity between cells encoding visual features and compass cells has been directly observed (Kim et al., 2019).

There are two main methods of using visual information to control the head direction bump position. The first is to use visual information to fine tune the model of AHV through a learning mechanism, whether that be by detecting error between the estimated head angle and the expected head angle based on the visual cue (Kreiser et al., 2020), or using a combination of strategic behavior and landmark tracking to match the AHV model to the movement of the cue within the visual field (Stratton et al., 2011). The second method is to influence the head direction bump position directly by exploiting the attractor dynamics and injecting current into the new bump position. This could simply use Gaussian inputs into the ring attractor at determined positions (Song and Wang, 2005), or by representing features in multiple "landmark bearing cells," learning the association between head angle and visible features, and feeding back expected head direction onto the head direction cells (Yan et al., 2021). A combination of these two methods of visual control is probably the answer to accounting for environmental or body changes in the real world; in this study we begin with directly influencing the bump position.

1.4. Models of Visual Input

In models of head direction stabilized by visual input, the visual data used is often contrived. For example, adopting "visual cells" that fire at specific head angles without any real visual data (Song and Wang, 2005) and assume visual processing is performed somewhere upstream. Where true vision is used (captured by cameras), one or more cues positioned in the environment, such as colored panels (Yan et al., 2021) or LEDs (Kreiser et al., 2020), are identified, mapped to a "visual cell," and learning mechanisms associate this cue with a head angle (Bicanski and Burgess, 2016). Natural visual scenes are much more complex and information rich than bold homogenous cues, with this richness making real-world landmark identification more difficult. The question remains, how does a visual scene become useful for maintaining head angle? In this work we use natural scenes, projected onto a sphere around a simulated robot (see **Figure 1**. We assume this visual information is distal and invariant to translation. We show that both generative and discriminative model-free learning algorithms can be used to predict head angle from natural visual information and correct for drift in a spiking continuous attractor model of head direction cells, without the need to identify specific landmarks in the environment.

1.5. On the Discriminative-Generative Dichotomy

Machine Learning approaches broadly draw from two paradigms. The first is a discriminative paradigm; models which aim to partition the incoming data samples along meaningful boundaries, often building a hierarchy of increasingly abstract representations or increasingly broad sub-spaces, extracting meaning through a bottom-up feedforward process. An example would be Decision Trees, which learn to partition data through recursive splitting on the data space by simple if-else rules (Breiman et al., 1984).

The second is a generative paradigm, which approaches the problem in the opposite direction. A generative model, often also a probabilistic model, aims to instead learn the capability of generating appropriate data samples like the training data in the appropriate contexts. Like a discriminative model, it tries to uncover abstract features in the data, but instead incorporates this into a model of latent features, refining its hypotheses about the underlying causes of the sensory data it is receiving. An example would be Gaussian Mixture Models, which model the problem space as a family of Gaussians with different parameter values (Reynolds, 2009).

Although details between each model vary considerably, the broad trend is that discriminative models are faster than their generative counterparts, but can only work within the bounds of the data they are provided. With the data space's dimensionality being potentially unlimited, this still provides a huge amount of capability, but a training set that does not adequately reflect the data space can lead to nonsensical outputs. Generative models, on the other hand, typically tend to be slower to categorize and slower to learn. However, by generating samples from a model of latent causes of the data, they are not limited by their inputs and can produce very different predictions from the data they are provided. For the case of well-defined and well-bounded problems, this is often surplus to requirements, but for many situations, such as with unfamiliar or incomplete data, this can be beneficial.

Many algorithms make use of elements of both. For example, the Variational Autoencoder (Kingma and Welling, 2014) has hidden layers that extract features from the data in a discriminative way, and use these features to train a multidimensional Gaussian space, the output of which is decoded by another discriminative layer stack to produce a sensible reconstruction of the input. A Generative Adversarial Network (Goodfellow et al., 2014) takes this even further, using a generative and discriminative network

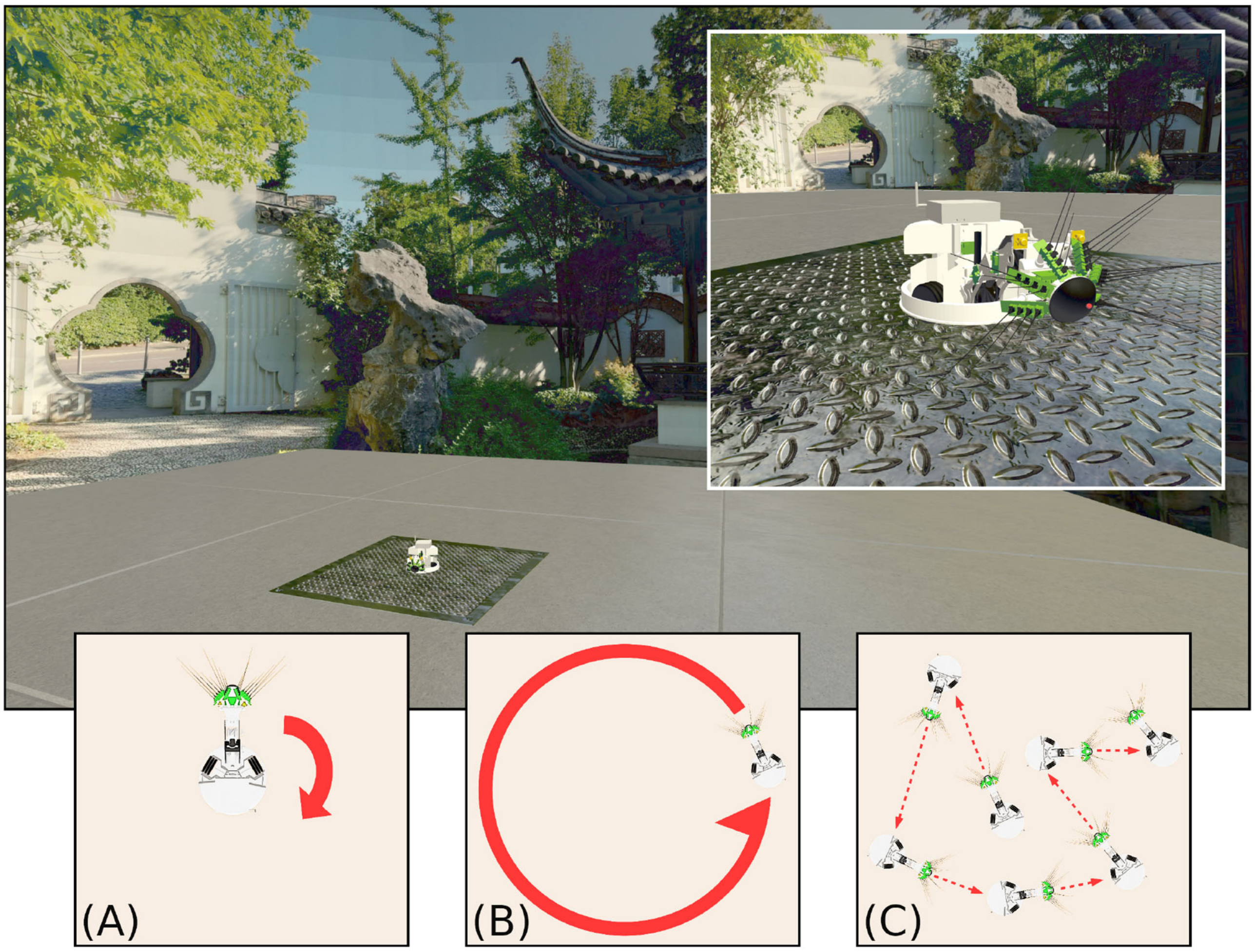


FIGURE 1 | The WhiskEye robot used to capture visual and odometry data sets, as it moved within the simulated environment of the NeuroRobotics Platform. The natural scenery (a panorama of the Chinese Garden in Stuttgart) was projected onto the inside surface of a sphere, surrounding a platform on which WhiskEye can move. The behaviours expressed by WhiskEye during capture of the data sets analysed in this study are referred to as **(A)** rotating, **(B)** circling, and **(C)** random walk.

in a collaborative competition to produce ever-better data samples. In neuroscience, particularly regarding the visual system, aspects of cortical function have been explained as both a discriminative and generative model, with exactly where and how these approaches synthesize together an active area of research (di Carlo et al., 2021); neural codes originally found in hippocampal work have been hypothesized as a unifying computational principle (Yu et al., 2021); see also Hawkins et al. (2019).

In this study we remain agnostic to the debate, instead choosing to evaluate a mix of generative and discriminative algorithms for generating predictive head direction signals from allothetic (visual) cues. As a purely generative model, a Predictive Coding Network based on MultiPredNet (Pearson et al., 2021), originally from Dora et al. (2018); as a hybrid model, a modification of the JMVAE from Suzuki et al. (2017); and as a purely discriminative model, a Convolutional Neural Network (Lecun and Bengio, 1997).

2. METHODS

2.1. Experimental Apparatus

WhiskEye is a rat-inspired omnidrive robot, with RGB cameras in place of eyes and an array of active whisker-like tactile sensors as shown in **Figure 1**. In this study only the visual frames from the left camera were considered. A simulated model of WhiskEye was integrated into the Human Brain Project's NeuroRobotics Platform (NRP) as part of prior work (Knowles et al., 2021; Pearson et al., 2021). The NRP integrates robot control and simulation tools, such as ROS and Gazebo, with neural simulators, such as NEST (Falotico et al., 2017). By running these in a synchronized way on a single platform, simulated robots can interact live with simulated neuron models, allowing for experiments with biomimetic and bio-inspired systems. Behaviors can also be specified in more controlled ways using the familiar ROS framework, whilst capturing data from both the robot and neural simulators for off-line analysis.

Using the NRP allows arbitrary visual scenes to be constructed within the environment. The visual scene in this experiment consisted of a concrete-textured floor for WhiskEye to move on; surrounded by an invisible collision mesh to contain the robot in the environment; and with an outer sphere to display the background. The sphere was made large enough so that translation had no perceptible effect on the visual scene; barring the concrete floor, all visual cues could be considered distal. Within this environment, WhiskEye executed three different behaviors: rotating on the spot, circling around the center of the environment and a random walk (illustrated in **Figures 1A–C**). This provided the odometry and visual data with which to validate the performance of each model.

2.2. Spiking Neural Network Model of Head Direction Cell System

The Head Direction system model is a spiking neural network (SNN) model written in pyNEST (2.18; Eppler et al., 2009). All cells are simulated using pyNEST's standard leaky integrate-and-fire neuron model (iaf_psc_alpha) which uses alpha-function shaped synaptic currents. The simulation timestep was set to 0.1 ms for high accuracy with synaptic delay of 0.1 ms. The network is composed of four equally sized rings of neurons: 180 Lateral Mammillary Nuclei (LMN) cells, 180 Dorsal Tegmental Nuclei (DTN) cells, 180 clockwise conjunctive cells and 180 anticlockwise conjunctive cells. Constant input current of 450 pA to all LMN neurons results in spontaneous firing at a rate of 50 spikes per second prior to inhibitory input from the DTN. A summary of the model can be found in **Table 1**.

Attractor dynamics emerge through reciprocal connections between cells in the excitatory LMN population and inhibitory DTN population. Each LMN cell e connects to a subset of DTN neurons with declining synaptic strength as a function of distance (**Figure 2A**). Reciprocal inhibitory connections from each DTN cell i to LMN cells are arranged with synaptic strength decreasing as a function of distance offset by a constant (μ). This arrangement provides inhibitory input to the cells surrounding the most active LMN cell, producing a single stable bump of activity.

LMN and DTN cells are arranged as rings for the purpose of defining synaptic strength based on distance; this gives the attractor network periodic boundaries. Distances between cells are found (D), accounting for the wrap around of the ring. Then synaptic weight from each LMN neuron to DTN neuron (W_{exc}), and return connections from DTN neurons to LMN neurons (W_{inh}) are defined as follows:

$$d1 = \left|\frac{e}{N_{ex}} - \frac{i}{N_{in}}\right|$$

$$d2 = \left|\frac{e}{N_{ex}} - \frac{i}{N_{in}} - 1\right|$$

$$d3 = \left|\frac{e}{N_{ex}} - \frac{i}{N_{in}} + 1\right|$$

TABLE 1 | Summary of the spiking neural network written using pyNEST.

Model Summary		
Neuron model	Standard pyNEST Leaky integrate-and-fire neuron model	
Synapse model	*static_synapse* does not support any kind of plasticity.	
Plasticity	-	
Topology	Populations arranged as rings	
Measurements	Spikes from LMN population	
Populations		
Name	**Size**	**N**
LMN	N_{ex}	180
DTN	N_{in}	180
CW conjunctive	N_{ex}	180
ACW conjunctive	N_{ex}	180
Connectivity		
LMN to DTN	$W_{exc} = b_{exc}exp\left(\frac{1}{2}\frac{-D^2}{\sigma^2}\right)$ where σ = 0.12 and b_{exc} = 4000	
DTN to LMN	$W_{inh} = b_{inh}exp\left(\frac{1}{2}\frac{-(D-\mu)^2}{\sigma^2}\right)$ where σ = 0.12 and μ = 0.5 and b_{inh} = 450	
LMN to CW conj	One to one, w = 660	
LMN to ACW conj	One to one, w = 660	
CW conj to LMN	$c[i]$ to $e[i+1]$, w = 169	
ACW conj to LMN	$c[i]$ to $e[i-1]$, w = 169	
Input		
AHV input	One *step_current_generator* per conjunctive cell population connected to all cells in the population.	
Allothetic input	One *step_current_generator* per cell in the LMN population connected one to one. Delivers current matching the prediction from the PCN, VAE or CNN.	

$$D = min\{d1, d2, d3\}$$

$$W_{exc} = b_{exc}exp\left(\frac{1}{2}\frac{-D^2}{\sigma^2}\right)$$

$$W_{inh} = b_{inh}exp\left(\frac{1}{2}\frac{-(D-\mu)^2}{\sigma^2}\right)$$

where e and i are the index of the excitatory and inhibitory cell, respectively. N_{ex} and N_{in} are the total number of cells in each ring. b_{exc} and b_{exc} are the base weight between the two populations. μ = 0.5 and σ = 0.12, which determine the position of the center of the peak, and the width of the peak, respectively.

In the absence of input from the two conjunctive cell populations, the bump of activity maintained by the attractor network remains stationary. The initial position of the activity bump is produced by applying a 300 pA step current for 100 ms to a nominal LMN cell.

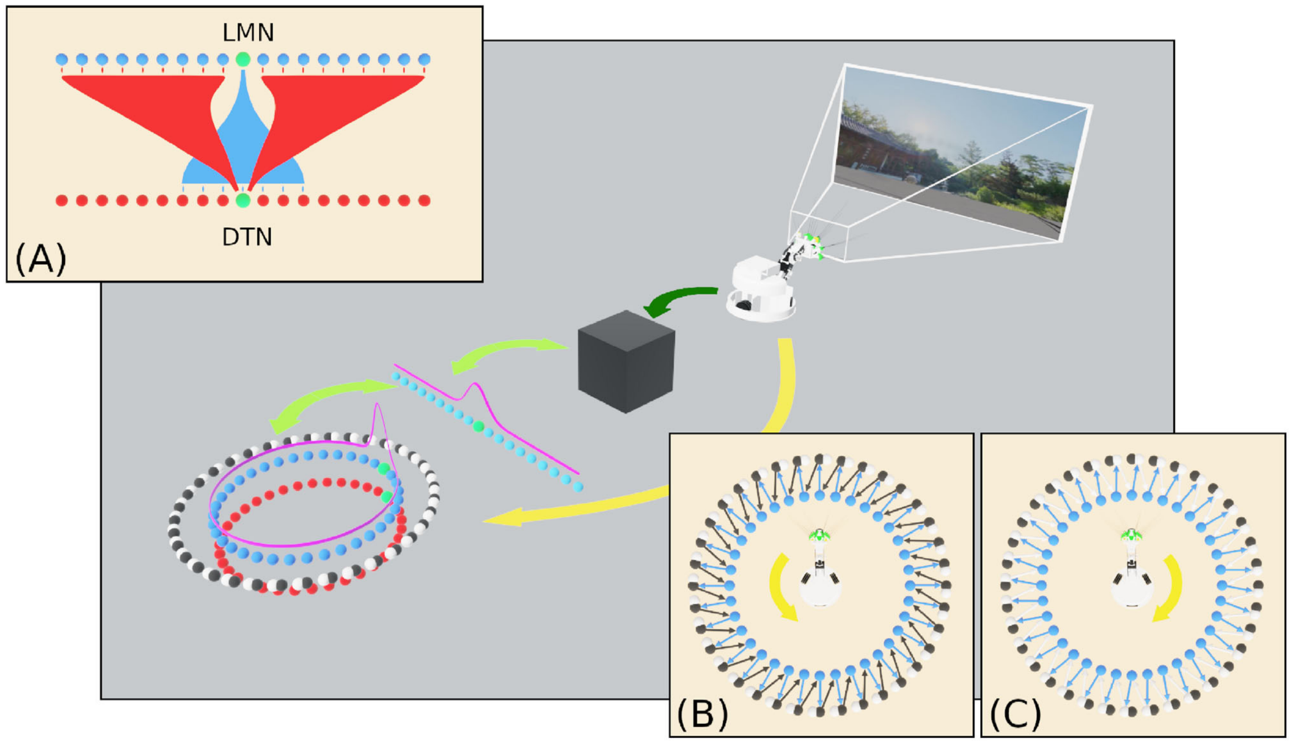


FIGURE 2 | An overview of the SNN model of head direction, the information flows and experimental setup of this study. Visual data from WhiskEye (dark green arrow) and the estimated head angle from the SNN (light green arrows) are used to train one of several model-free learning algorithms (black box). Once trained, these algorithms return head direction predictions (represented as the activity in the array of light blue neurons) that are mapped one-to-one with HD cells in the LMN ring (dark blue neurons) to correct for drift in the spiking HD ring attractor model. **(A)** Excitatory and inhibitory projections between the LMN and DTN respectively for the current most active cell (green neuron). Attractor dynamics emerge from this connectivity to maintain the bump in a stable position in the absence of idiothetic input. **(B)** Connectivity between anticlockwise conjunctive cells (black neurons) and head directions cells offset by one cell anticlockwise. With coincident head direction and angular velocity input (yellow arrow) these cells drive the bump clockwise around the ring. **(C)** Connectivity between clockwise conjunctive cells (white neurons) and head directions cells.

In order to track head direction based on the Angular Head Velocity (AHV) the two populations of conjunctive cells are connected one to one with a LMN cell, shifted one cell clockwise or anticlockwise from the equivalently positioned neuron (**Figures 2B,C**). Angular velocity of the head was determined by taking the first derivative of the head position captured from the simulated WhiskEye at a rate of 50 Hz, taking the difference in head angle at each time step. Positive values indicated anticlockwise head movements and negative values indicated clockwise head movements. AHV was converted to current (I_V) using the following formula:

$$I_V = (\theta_{t+1} - \theta_t) \cdot S + I_{min}$$

where θ is the head angle (radians) from the ROS topic published by the robot, $I_{min} = 150$ pA, and $S = 3500$. A step current generator supplies this current to the respective conjunctive cell population. LMN cells also connect one to one with the equivalent conjunctive cell in both the clockwise and anticlockwise populations. Spiking activity occurs in the conjunctive cells with coincident AHV and LMN spiking input. Conjunctive cell input causes movement of the attractor network activity bump to follow head movement.

2.2.1. Allothetic Correction

Head direction predictions from the visual learning models trained on Laplacian shaped representations of head direction (see below), are mapped one to one onto the respective head direction cells. Negative values in predictions are removed by adding the smallest value in the dataset, then prediction values are scaled by a factor of 10 and supplied to the HD network as a direct current injection. This simple method allows predictions which are smaller in magnitude to have less impact on the bump location. However, imprecise predictions, that may have multiple peaks or a broader shape, will lead to current input into more cells compared to a perfectly reconstructed Laplacian.

2.2.2. Analysis of Network Output

To compare the spiking network bump position to the ground truth, the most active cell in each 40 ms time window is found. The difference between the estimated head angle and the ground truth was used to show how accumulation of drift over time, with total error measured as Root Mean Squared Error (RMSE). To illustrate drift in the estimated head angle, the preferred firing direction was plotted using firing rate as a function of head angle in polar tuning curves. The ground truth head direction at each spike time was collected into bins (6°) for the first and the third minute, to show changes in preferred firing direction over time. Differences between idiothetic only and the three correction methods were compared using a one-way ANOVA combined with the Tukey HSD *post-hoc* test. A synthetic set of random uniform predictions, of the same shape and scale as the true predictions, were used to show that reductions in drift were not due to arbitrary current input. Statistical tests were performed using SPSS statistics software.

2.2.3. Artificial Cue Rotation

To investigate the control of allothetic cues over the head direction cell signal, we reproduced cue rotation experiments used in rodent studies (Taube and Burton, 1995). To supply current as if environmental cues were rotated by 90°, the predictions were manipulated by taking either the first 45 or 135 prediction values and shifting them to the end of the 180 element prediction, producing an artificial rotation. This rotation was applied for 30 s after 1 min of standard predictions.

2.3. Model-Free Learning Algorithms Applied to Allothetic Cue Recall

2.3.1. Dataset Preprocessing

Each dataset from WhiskEye contained both image data and head direction data. The image data processing was fairly simple, flattening each (width = 80, height = 45) RGB image into a single 10,800 long vector. The head direction data was more involved, being processed as follows:

- Head angle data was recorded at a much higher frequency of 50 Hz rather than the 5 Hz image data. It was therefore subsampled to match the timestamps of the image data.
- Head angle at each timestep from the odometry file was mapped to the 180 cell LMN structure. For example, a head angle of 120° would become a one-hot vector with the max at cell 60.
- In the case of the Spiking Neural Network Estimate, the most active cell in a given 40 ms window was chosen as the active cell for the head direction vector.
- A set of Laplacian distributions was created with means being the active cell of each head direction vector. The Laplacian was chosen over the conventional Gaussian as it lead to better performance overall for the three networks.
- These were rescaled so that the max value for each was 1.

2.3.2. Predictive Coding Network

This network was a modified version of the MultiPredNet (Pearson et al., 2021) which was developed for visuo-tactile place recognition. Here the 3 modules that made up the original network (visual, tactile and multisensory modules) were re-purposed as visual, head-direction and multisensory modules. Compared to the conventional feedforward architectures of other algorithms, the PCN relies on feedback connections toward the input data. For each sample, the PCN outputs a prediction from its latent layer that passes through the nodes of the hidden layers to the input later. The weights between each pair of layers transform the prediction from the upper layer into a prediction of the lower layer's activity. At each layer, the prediction from the layer above is compared to the activity at the current layer and the difference (error) calculated. Weights between layers are then updated locally according to their prediction errors. This eliminates the need for end-to-end backpropagation and increases bio-plausiblity. Several network topologies were trialed; the best performing network had direct odometry input into the multimodal latent layer, hence the lack of hidden layers in that stream (see **Table 2** for summary).

TABLE 2 | Model parameters and dataset details for PCN, VAE and CNN.

Parameter	Values		
	PCN	VAE	CNN
Visual input size	10,800	10,800	10,800
Visual hidden layers shape	[1,000, 300]	[1,000, 300]	[32(3,3), 64(3,3)]
Odometry input/output size	180	180	180
MSI layer shape	100	[50, 50]	N/A
Training epochs	200	5,000	50
Full set size	3,000	3,000	3,000
Single set size	390	390	390
SNN estimate size	390	390	390
Test set size	3,000	3,000	3,000
Validation set size	N/A	2,000	2,000
Learning rule	Hebbian	Backprop	Backprop
Optimiser	N/A	SGD	Adam

PCN training epochs are low due to number of inner 'cause epochs' (50 for training, 500 for inference) that increase training time, though are strictly a modification of the learning rule rather than extra training epochs. Note that the visual layers for the CNN are 2D Convolution Layers with the kernel shape in brackets. Training epochs vary but all networks were trained to convergence with a single epoch consisting of the entirety of the data for that dataset variant (full, single or SNN estimate). Validation data was taken from a separate set of data gathered with the WhiskEye Rotating behaviour. Hebbian learning is as per Dora et al. (2018), Backpropagation as per Chollet (2015).

2.3.3. Multi-Modal Variational Autoencoder

Based on the Suzuki et al. (2017) Joint Multimodal Variational Autoencoder, the VAE works by compressing inputs via hidden layers of decreasing size, encoding inputs into a bifurcated joint multimodal latent space representing the means and variances of Gaussians. These means and variances are used to generate normally distributed random variables, which are then passed through an expanding set of hidden layers to decode the latent Gaussian output into the same shape and structure as the input data. This encoder-decoder system is trained via conventional error backpropagation, comparing the decoded output to the 'ground truth' input and adjusting weights accordingly, with the addition of a KL-Divergence term to penalize divergence from a $\mu = 0$ Gaussian. As with PCN, the best performing network had no hidden layers between odometry input and the latent layers, so these were removed from both the encoder and decoder halves of the network.

2.3.4. Convolutional Neural Network

As a discriminative network, the CNN handles the task by training its weights so that a given visual input produces the corresponding correct head direction estimate Similarly, unlike the other two networks, the CNN has no latent space to condition and operates purely as a encoder, transforming visual scenes to their appropriate head direction output, with weights updated using conventional backpropagation. It is also the only network that is designed specifically for processing images, with strong spatial priors implicit in the way it processes visual scenes, analyzing small areas of the image in parallel via convolutions to produce translation-invariant image features. As the problem exists within a small, bounded space in both the visual and odometry domains for this experiment, the larger benchmark CNNs—AlexNet (Keshavarzi et al., 2021), ResNets (He et al., 2016) etc.—were not required. Instead, a lightweight, purpose-built CNN was created.

3. RESULTS

3.1. Head Direction Cell Like Firing Properties

Cells in the LMN, DTN and conjunctive cell populations all showed directional firing specificity as observed in the rodent brain. **Figure 3A** shows firing rate as a function of head direction from the equivalent cell in each of the LMN, DTN and conjunctive cell rings. The preferred firing direction of these cells is taken at the peak firing rate, and the directional firing range is the total range of angles each cell fires over. The average directional firing range of cells in the LMN was 59.3 ± 0.63°, DTN 275.7 ± 0.69° and conjunctive cells 59.2 ± 0.63° (**Figure 3B**). This is consistent with the directional firing range of DTN head direction cells in rodents (109.43 ± 6.84°; Sharp et al., 2001) being greater than the directional firing range of LMN head direction cells (83.4°; Taube et al., 1990). **Figure 3C** shows firing rate as a function of angular head velocity for an example conjunctive cell that has similar form to asymmetric AHV cells recorded in the DTN (Bassett and Taube, 2001).

3.2. Preferred Firing Direction of Cell Drift With Only Idiothetic Drive

Ring attractor dynamics which emerge from reciprocal connections between LMN and DTN cells maintain a stable bump of activity centered on the current estimate of head direction. When movement of the bump is driven only by idiothetic angular velocity input from the two conjunctive cell rings, the preferred direction of head direction cells drifted over time. **Figure 4A** shows the ground truth (black) and estimated

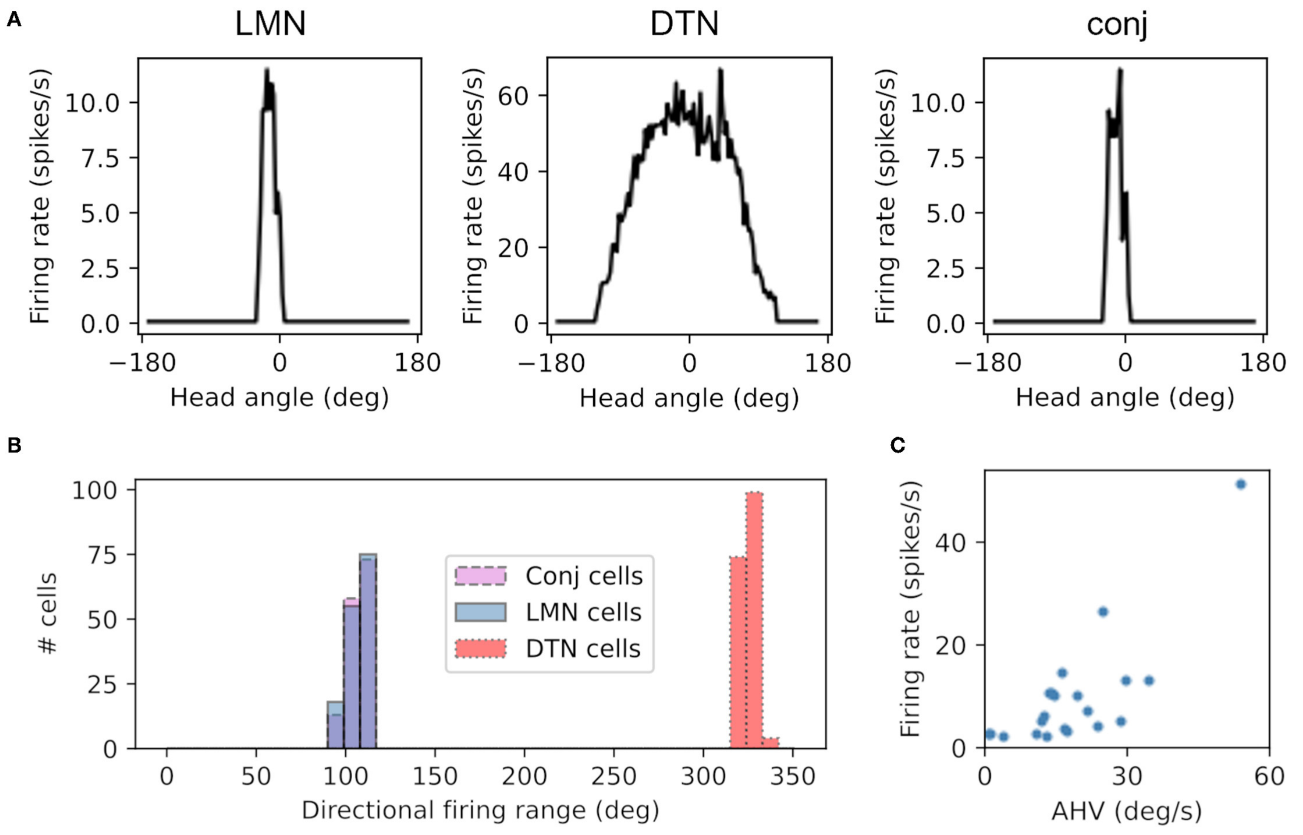


FIGURE 3 | Cells from the DTN, LMN and conjunctive cell populations show HD-like firing characteristics. **(A)** Preferred head angle of one DTN, LMN and conjunctive cell expressed as firing rate as a function of head angle, showing strong directional selectivity in the LMN and conjunctive cell and broader directional selectivity in the DTN cell. **(B)** Histogram of the directional firing range of cells in each population, showing broader directional firing in DTN cells. **(C)** Firing rate as a function of angular head velocity (AHV) from one example conjunctive cell.

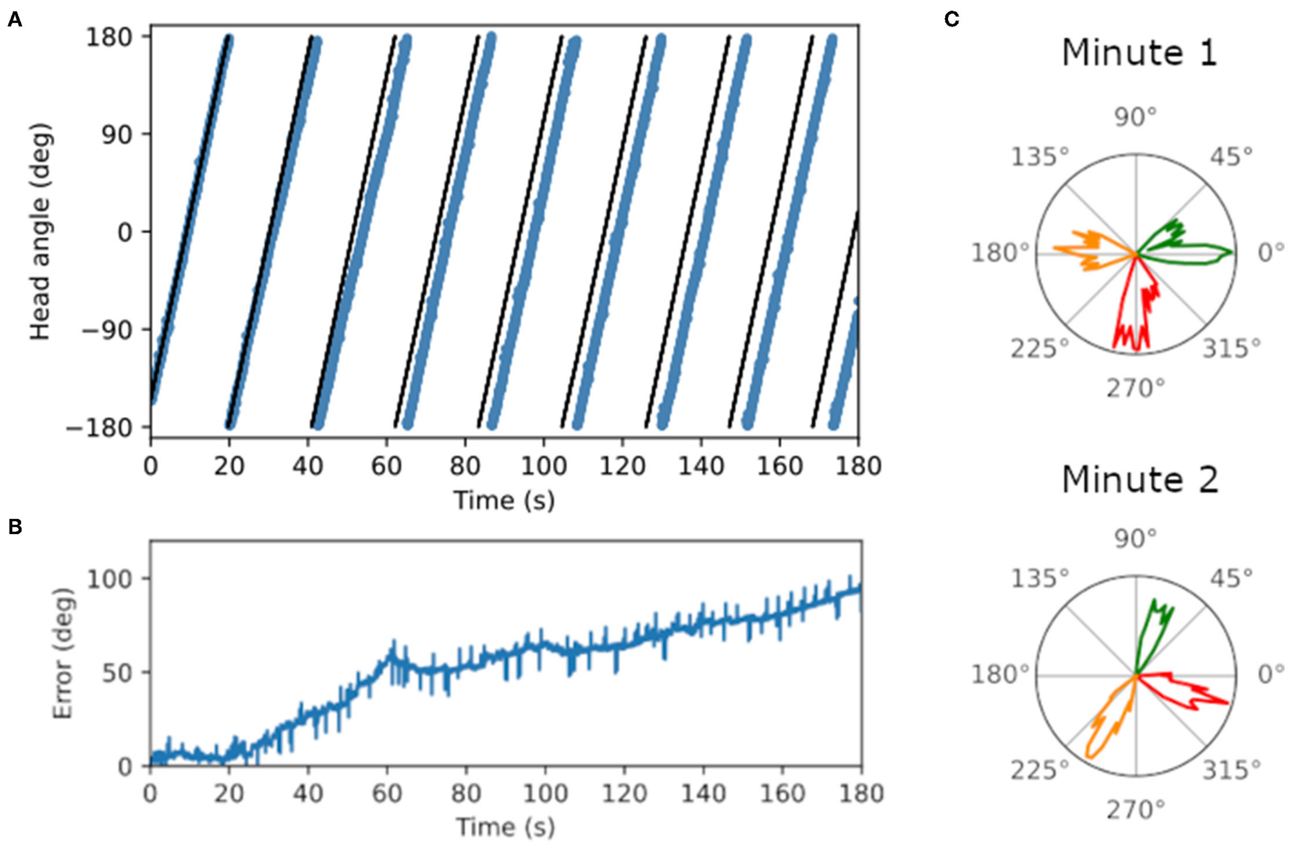


FIGURE 4 | Plots showing drift in the head direction estimate over time. **(A)** Ground truth head angle (black) and the estimated head angle (blue) from the SNN as the WhiskEye rotates on the spot. Over time the estimate gets further from the ground truth. **(B)** Error measured as the magnitude of the difference between the estimated angle and ground truth increases over time. **(C)** Preferred firing directions of three cells (red, orange and green) in the first vs third minute of the simulation, showing a change in preferred firing direction for all three cell of approximately 70°.

head direction (blue) over time when the WhiskEye robot rotates on the spot. The difference between the ground truth and estimate grows over time (**Figure 4B**), ending with a maximum difference of 94.5° after 3 min (RMSE = 58.4°). Firing rate as a function of time for 3 LMN cells in the first minute vs the third minute are shown in **Figure 4C**. The shift in preferred direction of these head direction cells from the first minute to the third minute was 51.3 ± 10.4°. However, the RMSE over the first full revolution was fairly low (5.2°).

3.3. Predicting Head Direction Using Model-Free Learning Algorithms

Rodents use allothetic information, such as vision, to counter this drift in head estimate. This requires forming associations between visual scenes and the current head angle, so that the estimated head angle can be corrected when this visual scene is experienced again. Ground truth head direction is not available in biology to form associations between visual scenes and heading. As drift in the head direction estimate (RMSE) is minimal during the first rotation (**Figure 4**), even when only idiothetic information is available, these early head direction estimates could be used for training the model-free learning algorithms. This would be a much smaller training set; to test the viability of using a such a reduced training set, we first used a single rotation of the ground truth.

This gave us three datasets to train on:

- Full Set - the full 3 min run of ground truth data
- Reduced Set - a single rotation of ground truth data
- SNN Estimate - a single rotation of idiothetic data

Head direction predictions made by three models trained on head direction/vision pairs are not equally structured. As seen in **Figure 5**, the discriminative CNN is far superior at generating a smooth Laplacian reconstruction, closely approximating the ground truth equivalent for all three variant datasets. The VAE reconstructions consisted of many competing peaks of varying heights, whilst the PCN shows qualities of both, maintaining a Laplacian-esque area of the distribution with noise increasing after a certain distance from ground truth head direction. Both generative models showed a noticeable degradation in the structure of their predictions on the smaller datasets; this is most apparent with the VAE, which suffered further degradation when trained with the SNN estimate.

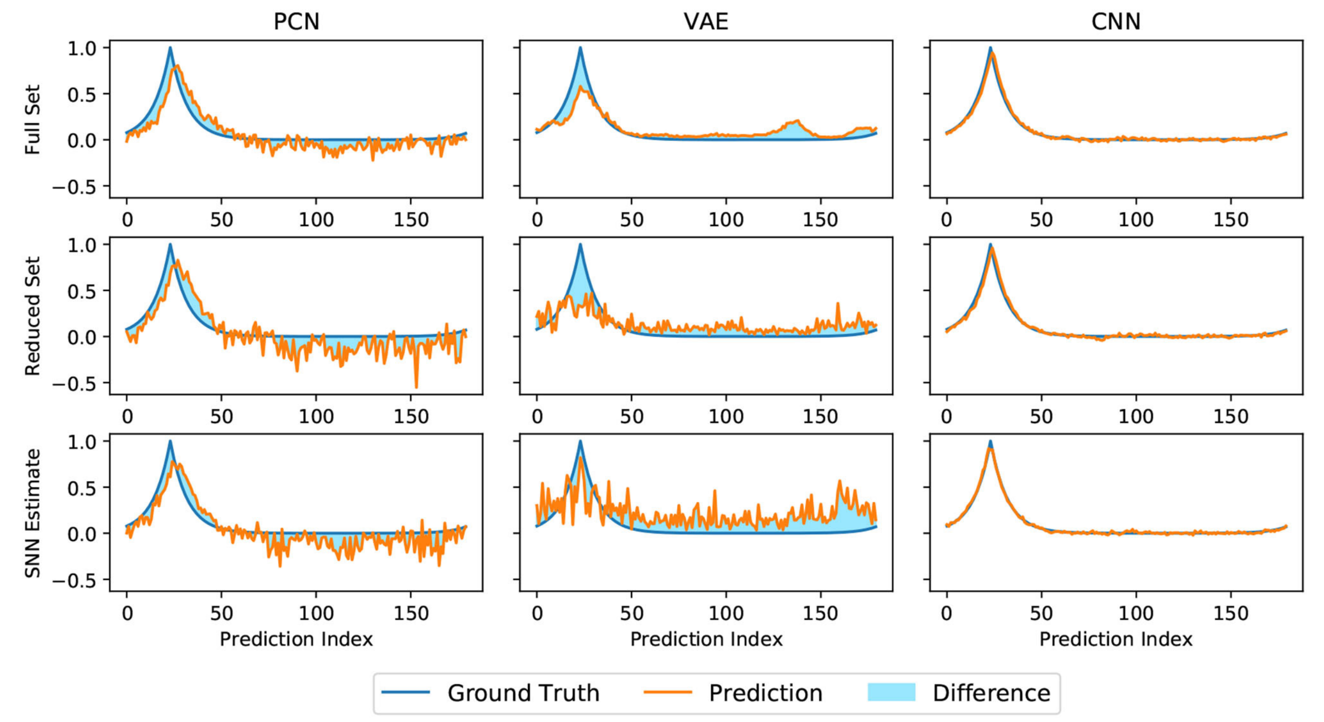


FIGURE 5 | Representative reconstructions of head direction predictions inferred by each algorithm (orange) at a given ground truth head direction (blue). The shaded difference between the two curves illustrates magnitude of the RMSE, which is inversely proportional to the quality of the reconstruction.

Figure 6 shows the reconstruction error (mean RMSE) for binned views of the visual scene taken from WhiskEye during the rotating behavior. Although there are variations in the error, the performance remains nominally uniform for all head angles, suggesting that the models are not favoring particular features for head direction estimate.

Figure 7 shows the overall reconstruction error (mean RMSE) for all datasets and scenarios. For all three models, reconstruction error was noticeably increased by a reduction in dataset quality, but the absolute error remains small. Both the reduced dataset and the SNN estimates were comparable in their error values, demonstrating that the internally generated estimates of the SNN model are a suitable substitute for ground truth odometry as a teaching signal, provided the dataset (and therefore accumulated drift) is small. Further to this, it demonstrates the effectiveness of all three methods, and thus their representative paradigms, at performing this task with limited data.

3.4. Drift Reduction Using Head Direction Predictions as Allothetic Input

The predictions generated by the PCN, VAE, and CNN trained on the full dataset were converted into one-to-one current inputs to LMN cells to correct for drift using visual information. **Figure 8** shows the ground truth head direction, idiothetic only estimate and the corrected estimate for 3 example datasets (rotation, a random walk, and circling), with the respective error over time. In each case, the corrected head direction estimate (pink) is much closer to the ground truth (black) than the estimate using idiothetic input only (blue), which drifts over time. Across all five random walk datasets, corrective input from the PCN, VAE and CNN all significantly reduced drift (one way ANOVA with Tukey HSD *post-hoc* testing: PCN $p = 0.001$; VAE $p < 0.001$; CNN $p < 0.001$, **Table 3**). The smallest error after corrections was achieved using predictions made by the CNN, which had the lowest reconstruction error. Even though VAE predictions are imprecise, it still performs comparably to the other methods. This may be due to current inputs onto HD cells far from the active bump having less influence due to the attractor dynamics; only current inputs close to the bump location have strong influence over bump position. Although the drift was large for the circling dataset (RMSE = 558.6°), all three methods successfully corrected for this drift. This was the biggest reduction in error for all three model-free learning algorithms (difference in RMSE: PCN 549.5°, VAE 555.2°, CNN 556.3°).

3.5. Drift Reduction Using a Reduced Training Set

The PCN, VAE, and CNN were trained using a single rotation of ground truth head directions, and the same method used to convert the predictions into current input to the head direction cells. In all cases, the RMSE between ground truth and the estimate head direction was reduced. Across all five random walk datasets, corrective input from the PCN, VAE and CNN all significantly reduced drift (one way ANOVA with Tukey HSD *post-hoc* testing: PCN $p = 0.001$; VAE $p = 0.002$; CNN $p < 0.001$, **Table 3**). Once

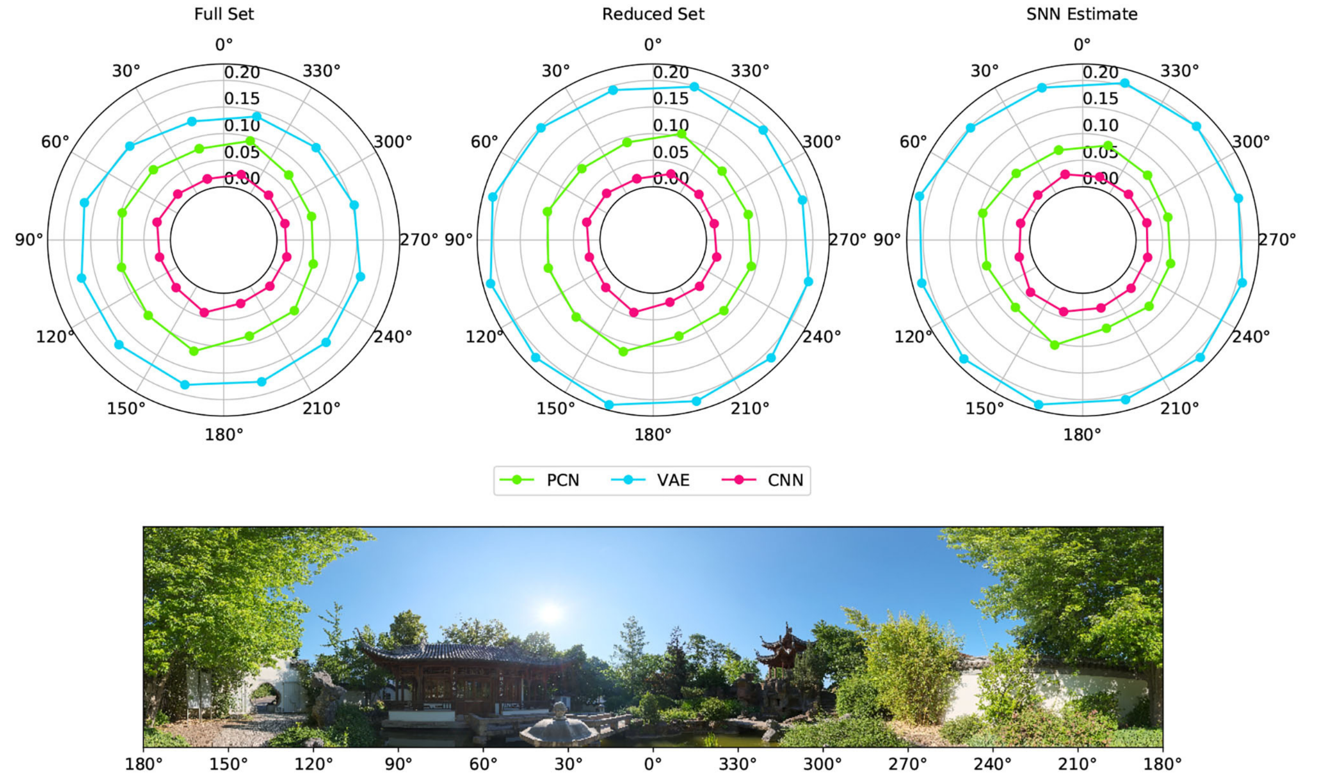


FIGURE 6 | Reconstruction error for different viewpoints of the environment taken from the rotating test dataset. The reconstruction error for each 30° arc of view is represented as a point with radial distance equal to the RMSE between the ground truth Laplacian and each model's reconstruction (PCN, VAE and CNN trained using each of the training sets). The panorama depicts the view of the robot as it rotates on the spot, with associated angular head direction labeled in register with the error polar plots above.

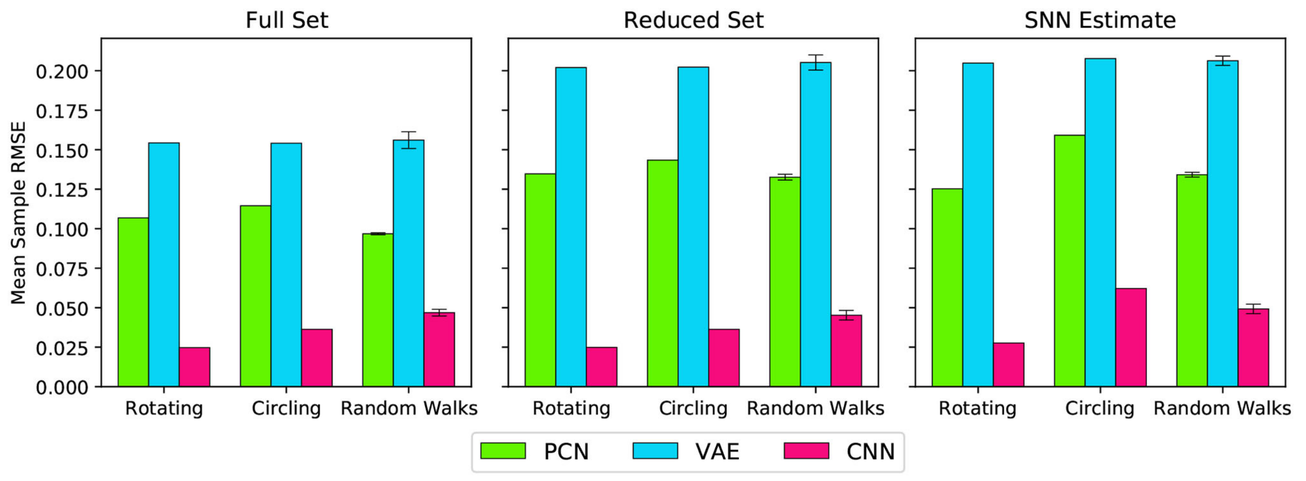


FIGURE 7 | Reconstruction error (RMSE) for each model, during each behaviour, trained on each training set. The Random Walks columns represent the mean of the 5 Random Walk datasets, with error bars indicating the standard error.

again the largest error reductions were achieved using CNN predictions. The VAE corrections were the least helpful, reflecting the larger reconstruction error when training on the reduced dataset.

3.6. Drift Reduction Using SNN Estimate as Training Set

As the reduction in drift was comparable when the full 3 min ground truth and a single revolution were used as training sets,

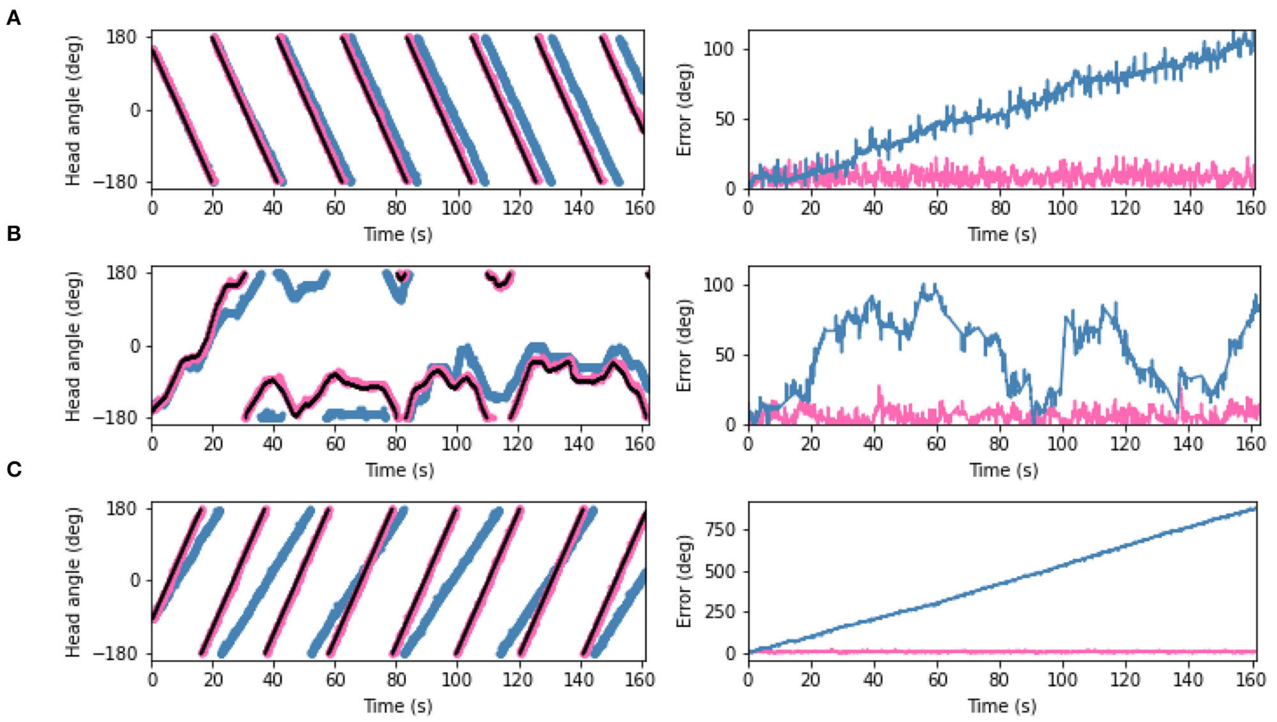


FIGURE 8 | (Left) Plots showing estimated head angle from the SNN with idiothetic drive only (blue), the corrected estimated head angle from the SNN which also receives allothetic input from the PCN (pink), and ground truth head angle (black). (Right) The difference between each estimate and the ground truth as also shown. Examples are shown from the **(A)** rotating, **(B)** random walk 1, and **(C)** circling datasets. In all cases, the allothetic correction results in minimised drift and the corrected estimate and ground truth are almost indistiguishable. As the PCN, VAE and CNN produce similar reductions in drift, only the PCN plots are shown.

TABLE 3 | RMSE (degrees) of the difference between the estimated head direction from the model and the ground truth using only idiothetic drive, and with corrections from the PCN, VAE or CNN trained on each of the three training sets.

		Full set RMSE (°)			**Reduced set RMSE (°)**			**SNN estimate RMSE (°)**		
	Ideo only	**PCN**	**VAE**	**CNN**	**PCN**	**VAE**	**CNN**	**PCN**	**VAE**	**CNN**
Rotation	69.64°	9.41°	3.44°	2.68°	9.41°	8.35°	2.31°	6.73°	6.99°	2.71°
Circling	558.81°	9.58°	3.72°	2.24°	14.21°	9.44°	2.47°	14.21°	14.86°	2.49°
Random 1	63.33°	7.02°	5.47°	4.24°	8.64°	15.70°	4.32°	9.02°	15.98°	3.22°
Random 2	58.18°	6.39°	4.71°	4.17°	8.73°	10.74°	4.64°	9.03°	18.97°	3.73°
Random 3	126.90°	7.25°	6.35°	2.61°	9.27°	18.47°	2.88°	10.21°	16.50°	4.05°
Random 4	203.09°	16.83°	13.21°	12.34°	14.36°	18.49°	10.62°	14.89°	16.83°	12.34°
Random 5	65.03°	7.09°	5.91°	3.35°	8.72°	12.08°	3.49°	8.42°	15.83°	3.57°

we trained each of the model-free learning algorithms on a single revolution of the estimated head direction produced by the spiking model.

Similar to the drift reduction seen for the previous two training sets, drift was reduced by all three models trained on each of the datasets. Across all five random walk datasets, corrective input from the PCN, VAE and CNN all significantly reduced drift (one way ANOVA with Tukey HSD *post-hoc* testing: PCN $p = 0.001$; VAE $p = 0.002$; CNN $p < 0.001$, **Table 3**). The CNN produced the best error reduction, ahead of the PCN and then the VAE, reflecting the reconstruction error of their predictions. With each decrease in training set quality from full ground truth, first to single revolution ground truth and then to single revolution estimated head direction, the average error across the random walks increased for both the PCN and VAE, remaining stable only for the CNN. **Figure 9** shows a summary of drift reduction by all three model-free learning algorithms trained on the full, reduced and SNN estimate training sets. Compared to head direction estimates which rely only on idiothetic input, or randomly generated predictions, all methods and training sets produced a significant reduction in drift.

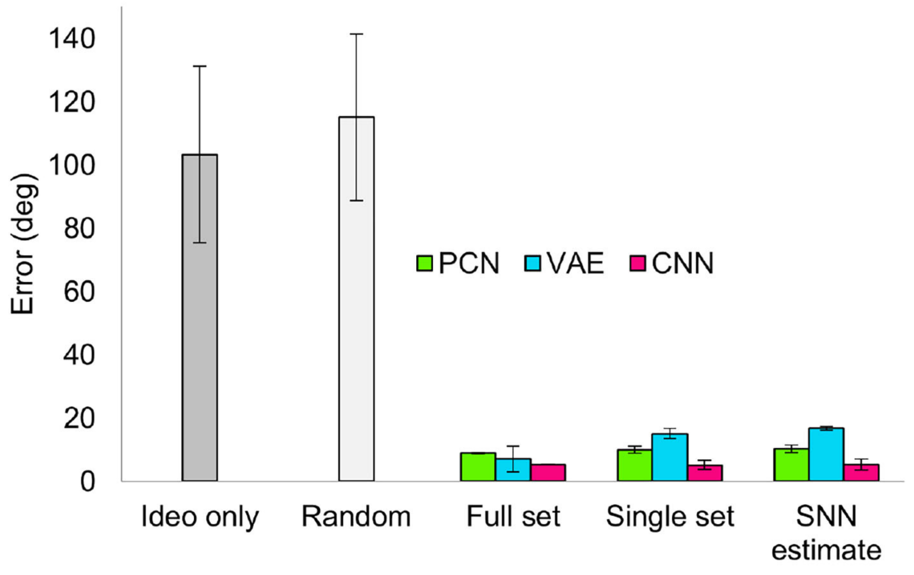


FIGURE 9 | Summary of drift reduction for each of the model-free learning algorithms and training sets across all 5 random walk datasets. Compared to idiothetic input only and random predictions, drift is significantly reduced for all three methods trained on full, single and SNN estimate training sets. Bar plot shows average error (degrees) ± standard error.

3.7. Cue Rotation

Head direction cells in rodents have been shown to follow environmental cues over their idiothetic estimate of heading, even when those cues are rotated within the environment (Taube and Burton, 1995; Yoder et al., 2015). To replicate a cue rotation experiment using the WhiskEye rotating behavior, we provided unaltered allothetic predictions from each of the model-free learning algorithms for the first minute, then rotated 90° either clockwise or anticlockwise for 30 s before returning to unaltered allothetic predictions. **Figure 10** shows the head direction estimate against the ground truth, with the error for clockwise (**Figure 10A**) and anti-clockwise (**Figure 10B**) rotations of the allothetic input from the PCN, VAE and CNN trained on the full ground truth. The green line shows an offset of 90°, which is the rotation of the cue and the value the error is expected to reach.

For both clockwise and anti-clockwise rotations, the PCN and VAE input strongly control the bump position. After a short delay, the bump position moves the full 90°, error between ground truth and the estimate reaching the green line. When the rotation is removed the bump continues to follow the allothetic input after a delay. Some drift may be required before the allothetic input can gain control over the bump position, resulting in a delay. In the case of the CNN, only when the idiothetic drive and the rotation were in the same direction (**Figure 10B**) could the allothetic input control the bump position strongly enough to complete the full rotation. Because the allothetic and idiothetic input are provided simultaneously, the bump is more likely to move when both of these pull the bump in the same direction around the ring rather than compete with each other.

The CNN has consistently the lowest reconstruction error of all three methods (**Figure 5**), producing predictions with a sharp Laplacian peak. This prediction shape results in current input to a small number of cells at a precise position, and produces the most accurate head direction estimate. This is likely because the amount of drift between each allothetic correction is small, and the bump does not need to be moved far. Noisier predictions from the VAE and PCN result in current injection to more cells, making it less accurate for drift correction but more able to move the bump large distances, as in this cue conflict case. These data suggest a refined Laplacian peak is not the most effective prediction shape for strong allothetic control over the head direction estimate. In all cases, the current magnitude used was high enough to correct for drift without impairing idiothetic control. By varying the amount of current supplied, allothetic input could have stronger or weaker control over the bump position regardless of prediction shape.

4. DISCUSSION

With these experiments we have shown that, like head direction cells recorded in rodents, a spiking continuous attractor model of head direction cells driven purely by self-motion (idiothetic) information is subject to drift. Taking inspiration from a number of previous studies (Boucheny et al., 2005; Song and Wang, 2005; Shipston-Sharman et al., 2016), we exploit reciprocal excitatory and inhibitory connections between the LMN and DTN to produce attractor dynamics which maintain a bump of activity at the estimated head angle.

Drift is thought to be caused by imprecise self-motion cues, but may also be due to inaccuracies in the model of angular head velocity (AHV). Variability within environments or the body,

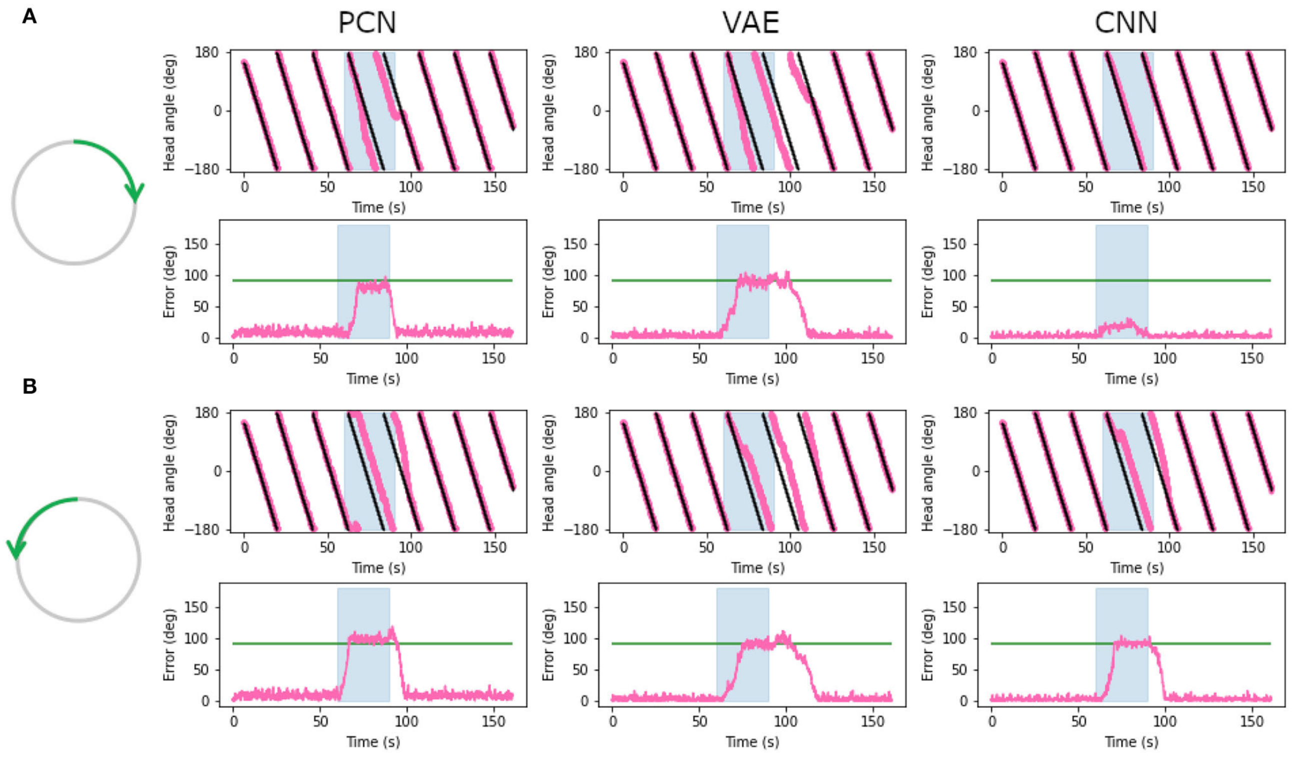


FIGURE 10 | Plots showing corrected estimated head angle (pink) compared the ground truth (black) during the artificial cue rotation experiments. The blue block indicates the period of cue rotation either clockwise **(A)** or anticlockwise **(B)**; the expected rotation (90°) is indicated with a green line on the error plot. In both clockwise and anticlockwise rotations, corrections by the PCN and the VAE move the bump to the rotated position after a delay. The CNN fails to pull the bump contrary to the direction of bump movement.

such as injury (or in robots; inaccuracy in the odometry data due to wheel slip), make maintaining a precise model of AHV at all times unlikely. A prominent limitation of the experimental apparatus is that the odometry from the robot being collected from a simulated embodiment is not subject to inaccuracies. However, the stochastic nature of a spiking model limits the resolution and range of angular velocities which can be accurately represented by a single neuron, this can be seen clearly in the large drift accrued during the circling dataset where head angle changes very slowly. Using a population code rather than single cells may allow for a finer resolution of AHVs which can be represented in spikes, and contribute to reducing drift.

In rodents, drift in the preferred firing directions of head direction cells is seen mainly in the dark or when brain regions providing allothetic input are lesioned (primarily visual; Yoder et al. 2015), indicating these data are essential for stabilizing the head direction signal. Using predictions from three different model-free learning algorithms, we directly influenced the bump position, minimizing drift. In some previous models of drift correction, allothetic information contributes to calibrating the model of AHV, rather than using allothetic input to directly change the bump position. Kreiser et al. (2020) refine the AHV model by detecting error between the estimated head angle and learnt positions of landmarks, and altering firing properties of AHV cells. Stratton et al. (2011) suggest a role for specific behaviour patterns for learning new landmarks and calibrating the AHV model. We show that predictions made after training on the estimated head angle from the SNN during a single revolution—a specific behavior—can be used to successfully correct for drift.

Entrainment of the head direction signal to visual information has been seen in cue rotation studies, where external environmental cues are rotated in the environment and a corresponding rotation is observed in the preferred firing direction of the head direction cells. These large changes in bump location are better solved by influencing the bump position directly, rather than updating the AHV model. By rotating the allothetic predictions, we have replicated shifts in the bump position to match the rotation of the environment. As AHV cell firing also shows some refinement when visual information is available (Keshavarzi et al., 2021), going forward a combination of optimizing the AHV model and direct bump movement could be used.

A Laplacian-shaped input centered on the current HD was used to train the three model-free learning algorithms. The CNN reproduced this shape in its prediction whereas the VAE and PCN produced broader, more Gaussian-like predictions. The CNN consistently produced the most precise head direction predictions even for the small SNN estimate training set. This suggests a trivial learning problem for the CNN, likely because the range of possible distal views observed by the robot is small and bounded; the same frames used for training are likely to be reobserved as the robot rotates. However, even with less precise predictions the PCN and VAE can reduce drift significantly, likely due to the attractor dynamics dampening current inputs far from the bump location. The artificial cue conflict experiments revealed a precise Laplacian distribution not to be suitable as a corrective signal due to the limited number of cells current is injected into, and therefore the limited power of this input to influence the bump location. In contrast, the broader predictions made by the PCN and the VAE were able to better control the bump position; refining the shape and strength of the prediction would likely change the allothetic control over the bump. Two populations of head direction cell have been identified in rodents, those more controlled by allothetic input and others more strongly controlled by idiothetic input (Dudchenko et al., 2019); we can see how by varying the strength or shape of allothetic input to the network, these two cell types may emerge.

In previous work, correcting drift with the aid of visual information has either assumed visual processing upstream and provided correction based on the ground truth (Song and Wang, 2005), or learnt the orientation of arbitrary features, such as LEDs or colored panels (Kreiser et al., 2020; Yan et al., 2021). Here we show that corrective signals can be generated by learning associations between natural visual scenes and a self generated representation of heading, without identifying specific environmental landmarks. However, we recognize that including advanced visual processing and feature extraction may be useful for online learning mechanisms to determine the reliability of visual input. This type of corrective allothetic signal is presumed to come from the postsubiculum; lesions of this region lead to more drift than seen for control animals in the dark (Yoder et al., 2015). This suggests that this region may be contributing more than just visual correction, but also other sensory modalities. In this paper, we have focused on the calibration of head direction estimate by visual inputs; an intriguing direction for future work would be the inclusion of other allothetic information, such as tactile or olfactory. In visually ambiguous environments, conflicting visual cues may cause the HD estimate to become less accurate. Olfaction has great potential for detecting loop closures, as rodents leave scent trails as the explore environments (Peden and Timberlake, 1990), which can tell them if and how long ago they visited a position. A recent study in mice has shown that blind animals can use olfactory information to correct for drift in the head direction estimate (Asumbisa et al., 2022).

All of the methods in this paper currently require batch learning of head direction-image pairs, however, as rodents continue to move within environments, they must continually learn and refine associations between head angle and visual scenes. Learning to place less weight on unreliable cues, such as the position of the sun, which may initially appear as a useful landmark but becomes unstable with time (Knight et al., 2014), is key to reliable correction of head angle in dynamic natural environments. The next step is to adapt these model-free learning algorithms to learn continuously and adapt their predictions as the robot explores its environment.

The three trained models, despite their differences in reconstruction error, are all good candidates for generating allothetic corrections for the SNN. Although some scenarios such as cue conflicts show weakness of overly-precise estimates as by the CNN, this is not a fault of the model itself; the robustness of the PCN and VAE predictions to cue conflicts shows that learning to minimize the RMSE from a Laplacian ground truth signal is not ideal for the task at hand, and that better performance could be gained by training to a broader distribution (such as a Gaussian).

Where differences do lie is in their applicability to more complex experimental setups. The environment the data is gathered from is simple in structure despite the complexity of the visual scene; there are no proximal cues to obscure the environment and sensory input is limited to vision. Previous studies have shown non- visual and multimodal examples of CNNs (Ma et al., 2015; Dauphin et al., 2017) and VAE architectures (Suzuki et al., 2017) can perform well. Both, however, have issues with scaling: the multimodal CNN requiring many stacked networks working together, and multimodal VAEs requiring many intermediate uni-modal latent spaces to perform the task successfully. It is an open question as to how well PCNs will scale into more than 2 modalities and whether they will run into similar scaling issues as the VAEs. However, its method of operation and learning rule are bio-plausible, with local learning making it the best candidate for implementation as a spiking model. Furthermore, prior work has already shown that a PCN can use tactile information to inform localization (Pearson et al., 2021).

This work has raised many important questions. How robust are these model-free learning approaches to a changing world, particularly with multiple environments, visually and potentially tactually distinct from each other? How can these be trained in a sequential manner, as an animal would experience them, whilst avoiding the catastrophic forgetting of earlier environments? Can the bio-plausibility of this system be increased by making the learning fully online, and is the SNN estimate of head direction reliable for long enough to train one of these algorithms to produce useful corrections? The experimental apparatus developed and used in this study are well placed to address these questions.

5. CONCLUSION

Idiothetic control of the head direction system is especially important in new, ambiguous or dark environments; allothetic control increases the accuracy of the head direction estimate and may help refine idiothetic control or make large corrections after a period of drift. We have shown that natural visual scenes, without identifying specific landmarks, can be used to predict the current head angle by training three model-free learning algorithms; this on a limited and imprecise training set of estimated head angles, produced by a spiking continuous attractor model of the head direction cell system, driven by idiothetic inputs from robot odometry. Predictions from all three methods were equally valuable in minimizing drift.

DATA AVAILABILITY STATEMENT

The datasets generated for this study, along with code for the NEST and Machine Learning models, can be found in the HeadDirectionPredNet repository: https://github.com/TomKnowles1994/HeadDirectionPredNet.

AUTHOR CONTRIBUTIONS

RS built the spiking neural network model. TK built the simulation experiments and gathered datasets. All authors contributed to the article and approved the submitted version.

ACKNOWLEDGMENTS

This research has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 945539 (Human Brain Project SGA3). Chinese Garden in Stuttgart panorama taken and released under CC0 license by Andreas Mitchok.

REFERENCES

Asumbisa, K., Peyrache, A., and Trenholm, S. (2022). Flexible cue anchoring strategies enable stable head direction coding in both sighted and blind animals. *bioRxiv*. 2022.01.12.476111. doi: 10.1101/2022.01.12.476111

Bassett, J. P., and Taube, J. S. (2001). Neural correlates for angular head velocity in the rat dorsal tegmental nucleus. *J. Neurosci*. 21, 5740–5751. doi: 10.1523/JNEUROSCI.21-15-05740.2001

Bassett, J. P., Wills, T. J., and Cacucci, F. (2018). Self-organized attractor dynamics in the developing head direction circuit. *Curr. Biol*. 28, 609.e3–615.e3. doi: 10.1016/j.cub.2018.01.010

Ben-Yishay, E., Krivoruchko, K., Ron, S., Ulanovsky, N., Derdikman, D., and Gutfreund, Y. (2021). Directional tuning in the hippocampal formation of birds. *Curr. Biol*. 31, 2592.e4–2602.e4. doi: 10.1016/j.cub.2021.04.029

Bicanski, A., and Burgess, N. (2016). Environmental anchoring of head direction in a computational model of retrosplenial cortex. *J. Neurosci*. 36, 11601–11618. doi: 10.1523/JNEUROSCI.0516-16.2016

Blair, H. T., Cho, J., and Sharp, P. E. (1999). The anterior thalamic head-direction signal is abolished by bilateral but not unilateral lesions of the lateral mammillary nucleus. *J. Neurosci*. 19, 6673–6683. doi: 10.1523/JNEUROSCI.19-15-06673.1999

Boucheny, C., Brunel, N., and Arleo, A. (2005). A continuous attractor network model without recurrent excitation: maintenance and integration in the head direction cell system. *J. Comput. Neurosci*. 18, 205–227. doi: 10.1007/s10827-005-6559-y

Breiman, L., Friedman, J. H., Olshen, R., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.

Cho, J., and Sharp, P. E. (2001). Head direction, place, and movement correlates for cells in the rat retrosplenial cortex. *Behav. Neurosci*. 115, 3–25. doi: 10.1037/0735-7044.115.1.3

Chollet, F. (2015). *Keras*. Available online at: https://keras.io.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). "Language modeling with gated convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70, ICML'17* (JMLR.org), 933–941.

di Carlo, J. J., Haefner, R., Isik, L., Konkle, T., Kriegeskorte, N., Peters, B., et al. (2021). "How does the brain combine generative models and direct discriminative computations in high-level vision?," in *Conference on Cognitive Computational Neuroscience*.

Dora, S., Pennartz, C., and Bohte, S. (2018). "A deep predictive coding network for inferring hierarchical causes underlying sensory inputs," in *International Conference on Artificial Neural Networks* (Rhodes: Springer), 457–467. doi: 10.1007/978-3-030-01424-7_45

Dudchenko, P. A., Wood, E. R., and Smith, A. (2019). A new perspective on the head direction cell system and spatial behavior. *Neurosci. Biobehav. Rev*. 105, 24–33. doi: 10.1016/j.neubiorev.2019.06.036

Eppler, J. M., Helias, M., Muller, E., Diesmann, M., and Gewaltig, M. O. (2009). Pynest: a convenient interface to the nest simulator. *Front. Neuroinf*. 2, 2008. doi: 10.3389/neuro.11.012.2008

Falotico, E., Vannucci, L., Ambrosano, A., Albanese, U., Ulbrich, S., Tieck, J. C. V., et al. (2017). Connecting artificial brains to robots in a comprehensive simulation framework: the neurorobotics platform. *Front. Neurorobot*. 11, 2. doi: 10.3389/fnbot.2017.00002

Fisher, Y. E., Lu, J., D'Alessandro, I., and Wilson, R. I. (2019). Sensorimotor experience remaps visual input to a heading-direction network. *Nature* 576, 121–125. doi: 10.1038/s41586-019-1772-4

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. *Adv. Neur. Infm Process. Syst*. 27.

Goodridge, J. P., Dudchenko, P. A., Worboys, K. A., Golob, E. J., and Taube, J. S. (1998). Cue control and head direction cells. *Behav. Neurosci*. 112, 749–761. doi: 10.1037/0735-7044.112.4.749

Goodridge, J. P., and Taube, J. S. (1995). Preferential use of the landmark navigational system by head direction cells in rats. *Behav. Neurosci*. 109, 49–61. doi: 10.1037/0735-7044.109.1.49

Grieves, R. M., and Jeffery, K. J. (2017). The representation of space in the brain. *Behav. Processes*. 135, 113–131. doi: 10.1016/j.beproc.2016.12.012

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., and Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature* 436, 801–806. doi: 10.1038/nature03721

Hawkins, J., Lewis, M., Klukas, M., Purdy, S., and Ahmad, S. (2019). A framework for intelligence and cortical function based on grid cells in the neocortex. *Front. Neural Circ*. 12, 121. doi: 10.3389/fncir.2018.00121

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV: IEEE), 770–778.

Keshavarzi, S., Bracey, E. F., Faville, R. A., Campagner, D., Tyson, A. L., Lenzi, S. C., et al. (2021). Multisensory coding of angular head velocity in the retrosplenial cortex. *Neuron* 110, 532.e9–543.e9. doi: 10.2139/ssrn.3821958

Kim, S. S., Hermundstad, A. M., Romani, S., Abbott, L. F., and Jayaraman, V. (2019). Generation of stable heading representations in diverse visual scenes. *Nature* 576, 126–131. doi: 10.1038/s41586-019-1767-1

Kim, S. S., Rouault, H., Druckmann, S., and Jayaraman, V. (2017). Ring attractor dynamics in the drosophila central brain. *Science* 356, 849–853. doi: 10.1126/science.aal4835

Kingma, P., and Welling, M. (2014). "Auto-encoding variational bayes," in *International Conference on Learning Representations* (Banff).

Knight, R., Piette, C. E., Page, H., Walters, D., Marozzi, E., Nardini, M., et al. (2014). Weighted cue integration in the rodent head direction system. *Philos. Trans. R. Soc. B Biol. Sci*. 369, 20120512. doi: 10.1098/rstb.2012.0512

Knowles, T. C., Stentiford, R., and Pearson, M. J. (2021). "WhiskEye: A biomimetic model of multisensory spatial memory based on sensory reconstruction," in *Annual Conference Towards Autonomous Robotic Systems* (Springer), 408–418. doi: 10.1007/978-3-030-89177-0_43

Kreiser, R., Waibel, G., Armengol, N., Renner, A., and Sandamirskaya, Y. (2020). "Error estimation and correction in a spiking neural network for map formation in neuromorphic hardware," in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (Paris: IEEE), 6134–6140.

Krichmar, J. L., Severa, W., Khan, M. S., and Olds, J. L. (2019). Making bread: Biomimetic strategies for artificial intelligence now and in the future. *Front. Neurosci*. 13, 666. doi: 10.3389/fnins.2019.00666

Kropff, E., Carmichael, J. E., Moser, M. B., and Moser, E. I. (2015). Speed cells in the medial entorhinal cortex. *Nature* 523, 419–424. doi: 10.1038/nature14622

Lecun, Y., and Bengio, Y. (1997). "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, Vol. 3361.

Lever, C., Burton, S., Jeewajee, A., O'Keefe, J., and Burgess, N. (2009). Boundary vector cells in the subiculum of the hippocampal formation. *J. Neurosci*. 29, 9771–9777. doi: 10.1523/JNEUROSCI.1319-09.2009

Ma, L., Lu, Z., Shang, L., and Li, H. (2015). "Multimodal convolutional neural networks for matching image and sentence," in *2015 IEEE International Conference on Computer Vision (ICCV)* (Santiago: IEEE), 2623–2631.

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., and Moser, M. B. (2006). Path integration and the neural basis of the 'cognitive map'. *Nat. Rev. Neurosc*. 7, 663–678. doi: 10.1038/nrn1932

O'Keefe, J. (1976). Place units in the hippocampus of the freely moving rat. *Exp. Neurol*. 51, 78–109. doi: 10.1016/0014-4886(76)90055-8

Page, H. J., and Jeffery, K. J. (2018). Landmark-based updating of the head direction system by retrosplenial cortex: a computational model. *Front. Cell Neurosci*. 12, 191. doi: 10.3389/fncel.2018.00191

Pearson, M., Dora, S., Struckmeier, O., Knowles, T., Mitchinson, B., Tiwari, K., et al. (2021). Multimodal representation learning for place recognition using deep hebbian predictive coding. *Front. Rob. AI* 8, 403. doi: 10.3389/frobt.2021.732023

Peden, B. F., and Timberlake, W. (1990). Environmental influences on flank marking and urine marking by female and male rats (rattus norvegicus). *J. Comparat. Psychol*. 104, 122–130. doi: 10.1037/0735-7036.104.2.122

Reynolds, D. A. (2009). Gaussian mixture models. *Encyclo. Biomet*. 741, 659–663.

Sharp, P. E., Tinkelman, A., and Cho, J. (2001). Angular velocity and head direction signals recorded from the dorsal tegmental nucleus of gudden in the rat: implications for path integration in the head direction cell circuit. *Behav. Neurosci*. 115, 571–588. doi: 10.1037/0735-7044.115.3.571

Shipston-Sharman, O., Solanka, L., and Nolan, M. F. (2016). Continuous attractor network models of grid cell firing based on excitatory-inhibitory interactions. *J. Physiol*. 594, 6547–6557. doi: 10.1113/JP270630

Song, P., and Wang, X. J. (2005). Angular path integration by moving "Hill of Activity": a spiking neuron model without recurrent excitation of the head-direction system. *J. Neurosci*. 25, 1002–1014. doi: 10.1523/JNEUROSCI.4172-04.2005

Stackman, R. W., Golob, E. J., Bassett, J. P., and Taube, J. S. (2003). Passive transport disrupts directional path integration by rat head direction cells. *J. Neurophysiol*. 90, 2862–2874. doi: 10.1152/jn.00346.2003

Stackman, R. W., and Taube, J. S. (1998). Firing properties of rat lateral mammillary single units: head direction, head pitch, and angular head velocity. *J. Neurosci*. 18, 9020–9037. doi: 10.1523/JNEUROSCI.18-21-09020.1998

Stratton, P., Milford, M., Wyeth, G., and Wiles, J. (2011). Using strategic movement to calibrate a neural compass: A spiking network for tracking head direction in rats and robots. *PLoS ONE* 6, 25687. doi: 10.1371/journal.pone.0025687

Suzuki, M., Nakayama, K., and Matsuo, Y. (2017). "Joint multimodal learning with deep generative models," in *International Conference on Learning Representations* (Toulon).

Tan, H. M., Bassett, J. P., O'Keefe, J., Cacucci, F., and Wills, T. J. (2015). The development of the head direction system before eye opening in the rat. *Curr. Biol.* 25, 479–483. doi: 10.1016/j.cub.2014.12.030

Taube, J. S. (1995). Head direction cells recorded in the anterior thalamic nuclei of freely moving rats. *J. Neurosci*. 15, 70–86. doi: 10.1523/JNEUROSCI.15-01-00070.1995

Taube, J. S., and Burton, H. L. (1995). Head direction cell activity monitored in a novel environment and during a cue conflict situation. *J. Neurophysiol*. 74, 1953–1971. doi: 10.1152/jn.1995.74.5.1953

Taube, J. S., Muller, R. U., and Ranck, J. B. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. I. description and quantitative-analysis. *J. Neurosci*. 10, 420–435. doi: 10.1523/JNEUROSCI.10-02-00420.1990

Vann, S. D., Aggleton, J. P., and Maguire, E. A. (2009). What does the retrosplenial cortex do? *Nat. Rev. Neurosci*. 10, 792–802. doi: 10.1038/nrn2733

Vinepinsky, E., Cohen, L., Perchik, S., Ben-Shahar, O., Donchin, O., and Segev, R. (2020). Representation of edges, head direction, and swimming kinematics in the brain of freely-navigating fish. *Sci. Rep*. 10, 1–16. doi: 10.1038/s41598-020-71217-1

Yan, Y., Burgess, N., and Bicanski, A. (2021). A model of head direction and landmark coding in complex environments. *PLoS Comput. Biol*. 17, e1009434. doi: 10.1371/journal.pcbi.1009434

Yoder, R. M., Clark, B. J., and Taube, J. S. (2011). Origins of landmark encoding in the brain. *Trends Neurosci*. 34, 561–571. doi: 10.1016/j.tins.2011.08.004

Yoder, R. M., Peck, J. R., and Taube, J. S. (2015). Visual landmark information gains control of the head direction signal at the lateral mammillary nuclei. *J. Neurosci*. 35, 1354–1367. doi: 10.1523/JNEUROSCI.1418-14.2015

Yoder, R. M., and Taube, J. S. (2009). Head direction cell activity in mice: robust directional signal depends on intact otolith organs. *J. Neurosci*. 29, 1061–1076. doi: 10.1523/JNEUROSCI.1679-08.2009

Yoder, R. M., and Taube, J. S. (2014). The vestibular contribution to the head direction signal and navigation. *Front. Integr. Neurosci*. 8, 32. doi: 10.3389/fnint.2014.00032

Yu, L. Q., Park, S. A., Sweigart, S. C., Boorman, E. D., and Nassar, M. R. (2021). "Do grid codes afford generalization and flexible decision-making?," in *Conference on Cognitive Computational Neuroscience*.

Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory. *J. Neurosci*. 16, 2112–2126. doi: 10.1523/JNEUROSCI.16-06-02112.1996

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

ORIGINAL RESEARCH
published: 01 June 2022
doi: 10.3389/fnbot.2022.850013

The Curved Openspace Algorithm and a Spike-Latency Model for Sonar-Based Obstacle Avoidance

Chenxi Wen[1] *and Timothy K. Horiuchi*[1,2]*

[1] *Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, United States,*
[2] *Neuroscience and Cognitive Sciences Program, University of Maryland, College Park, MD, United States*

The rapid control of a sonar-guided vehicle to pursue a goal while avoiding obstacles has been a persistent research topic for decades. Taking into account the limited field-of-view of practical sonar systems and vehicle kinematics, we propose a neural model for obstacle avoidance that maps the 2-D sensory space into a 1-D motor space and evaluates motor actions while combining obstacles and goal information. A two-stage winner-take-all (WTA) mechanism is used to select the final steering action. To avoid excessive scanning of the environment, an attentional system is proposed to control the directions of sonar pings for efficient, task-driven, sensory data collection. A mobile robot was used to test the proposed model navigating through a cluttered environment using a narrow field-of-view sonar system. We further propose a spiking neural model using spike-timing representations, a spike-latency memory, and a "race-to-first-spike" WTA circuit.

Keywords: attention, winner-take-all, bat echolocation, neural model, spike latency, collision avoidance, robotics

Edited by:
Feihu Zhang,
Northwestern Polytechnical University, China

Reviewed by:
Yanqing Chen,
Johnson & Johnson Pharmaceutical Research and Development, United States
Xinxing Chen,
Southern University of Science and Technology, China

Correspondence:
Timothy K. Horiuchi
timmer@umd.edu

Received: *07 January 2022*
Accepted: *04 May 2022*
Published: *01 June 2022*

Citation:
Wen C and Horiuchi TK (2022) The Curved Openspace Algorithm and a Spike-Latency Model for Sonar-Based Obstacle Avoidance.
Front. Neurorobot. 16:850013.
doi: 10.3389/fnbot.2022.850013

INTRODUCTION

Traveling through an environment toward a goal without colliding with obstacles is one of many essential abilities for animals to survive. Animals are often able to detect obstacles using different types of sensors to quickly decide on the motions to avoid them. In addition, animals are often observed to orient their heads in different directions to gather sensory information needed for obstacle avoidance. In the world of robotics, there have historically been two extreme philosophical starting points in the approach to solving this problem: rigorous path planning assuming accurate and extensive sensing (Latombe, 2012) and fast reflexive behaviors based on minimal and unsophisticated sensing (Braitenberg, 1986). Clearly, there is an expansive world of algorithms lying between these two extremes. Path planning algorithms calculate routes between starting and goal points, requiring extensive knowledge of the environment and accurate localization. These are appropriate when a tremendous amount of relevant knowledge about the world is available and optimal paths are desired. In contrast, reflexive algorithms simply steer the creature away from obstacles upon detection with very little latency (Milde et al., 2017). Although reflexive behaviors are well-suited to a creature traveling quickly through an unknown or changing sparse environment, even mildly cluttered environments can produce inappropriate movements. Philosophically, obstacles should not determine the direction in which a creature should move, rather they should simply indicate where the creature should *not* go. The question is then, "given the information about multiple sensed obstacles and the target location, how do we combine them to select a good path?"

Echolocating bats are excellent examples of creatures that possess such a capability. They predominantly use ultrasonic echoes to perceive their surroundings and fly through dense forests in complete darkness with ease. During their hunt for flying insects or other prey, they can avoid obstacles while homing in on their prey at the same time. Big brown bats (*Eptesicus fuscus*), using sonar calls that last 2–3 ms, send out sonar pulses to detect obstacles at a rate of up to 90 Hz and have been shown to fly amongst obstacles at a flight speed of 2–5 m/s in indoor environments (Sändig et al., 2014). Big brown bats are also observed to turn their heads to ping in different directions to gather sensory information while flying in a field of obstacles (Surlykke et al., 2009). Understanding how biological systems can deftly transform the storm of sensory information into motor actions to pursue a goal while avoiding obstacles has long been a goal for engineers and neuroscientists.

Sonar has several advantages compared to other sensing domains (e.g., vision, LIDAR, or infrared sensors) commonly utilized in animals or robot systems. For example, sonar has the capability of penetrating smoke or fog where LIDAR, infrared sensors or cameras can struggle. Sonar also works in all light conditions, whereas it can often be difficult for LIDAR and infrared sensors to work in bright lighting, and cameras often struggle to work in complete darkness. Regardless of the sensing modality, once obstacles have been detected, there are many proposed approaches to this local obstacle-avoidance problem for a robot or a vehicle. One popular set of approaches is based on vector summation. In these approaches, obstacles create repulsive force fields and goals create attractive force fields. The summation of these forces steers the vehicle along a safe path without colliding with obstacles. Some specific implementations are APF (Artificial Potential Fields) (Khatib, 1986; Lyu and Yin, 2019; Rostami et al., 2019; Shin and Kim, 2021) and VFF (Vector Field Force) (Borenstein and Koren, 1989). These algorithms are effective and interesting in their computational simplicity and mathematical elegance. They have problems such as the vehicle being trapped in local minima, although recent modifications have been proposed (Rostami et al., 2019) to solve the local minima problem. As mentioned earlier, however, we believe that obstacles should not turn the vehicle in any particular direction but only indicate where it should not go.

Other approaches to navigation on a planar floor divide the surroundings of the vehicle into angular sectors and transform them into a polar histogram. In this histogram, the proximity of obstacles in each sector is represented and the next direction in which to steer is calculated based on the values of the histogram. These approaches include VFH (Vector Field Histogram) (Borenstein and Koren, 1991; Wu et al., 2020), its extension VFH+ (Vector Field Histogram Plus) (Ulrich and Borenstein, 1998) and the "Openspace" algorithm (Horiuchi, 2009). These approaches are good for local maneuvering but are centered in the sensory domain and do not generally consider vehicle kinematics. There are also velocity methods that map the Cartesian space into the velocity space that represents the linear and angular velocities of the vehicle, then calculate the next movement of the vehicle in the velocity space. These methods are suitable for differential or holonomic vehicles because a point in the velocity space corresponds to a velocity that is directly executable on the vehicle. The algorithm evaluates a range of possible velocities in the velocity space according to an objective function that includes criteria such as speed, distances of the obstacles, or goal direction. Some specific implementations include CVM (Curvature Velocity Method) (Simmons, 1996; Molinos et al., 2014), DWA (Dynamic Window Approach) (Fox et al., 1997), and its recent extension DW4DO (Dynamic Window for Dynamic Obstacles) (Molinos et al., 2019).

There are collision avoidance algorithms that rely on deliberate planning (Aggarwal and Kumar, 2020; Yasin et al., 2020). In these algorithms, an optimal or near-optimal path with collision-free routes is calculated based on an environmental map that the vehicle senses and updates. These algorithms typically assume extensive, accurate maps over which long trajectories are tested sequentially, requiring both significant computational resources and memory as well as fast, accurate sensing. To address the high computational complexity of these algorithms, several optimization methods have been developed. In Pérez-Carabaza et al. (2019), a minimal time search algorithm with ant colony optimization is used to calculate the optimal path under communication-related constraints. The algorithm in Bry and Roy (2011) incrementally constructs a graph of trajectories while efficiently searching over candidate paths, resulting in a search tree in belief space that converges to the optimal path. Using this algorithm, aggressive flight of a fixed-wing air vehicle in an unstructured 3D environment was demonstrated (Bry et al., 2012). Maintaining and updating a metrically correct spatial map, however, is difficult to implement in a biologically plausible neural system.

The Openspace algorithm proposed in Horiuchi (2009) provides a neuromorphic VLSI implementation of the sensory-oriented histogram approach with a latency-based, spiking neural network, giving insights into how a biological system might implement sonar-based navigation. The Openspace algorithm seeks to find the most desirable straight-line direction of travel to avoid obstacles (**Figure 1**) while a bat is traveling on a 2-D plane. It divides the area in front into a number of steering directions, evaluates their desirability, and selects the winning direction with the maximum evaluation. It combines different inputs (a goal direction and detected obstacles) into a decision function to determine the steering decision. To physically travel precisely in the selected direction, however, a bat must be capable of making extremely sharp turns. In practice, a flying bat can only rotate gradually to the desired direction. This produces an overshoot that will require ongoing corrections that lead to a mismatch between the selected path and the actual path the bat flies on. In contrast, our proposed algorithm, the Curved Openspace Algorithm, projects the sensory-based obstacle data into "motor coordinates" before comparing motor choices similar to the velocity approaches (Simmons, 1996; Fox et al., 1997; Molinos et al., 2014). For a flying bat, this could mean selecting different turning radii (i.e., circular trajectories) as illustrated in **Figure 2**.

Another limitation of the original Openspace algorithm is the assumption that the bat has a wide field-of-view (FOV) that covers all directions with the same effective range, thus localizing all obstacles in front of the animal with a single sonar

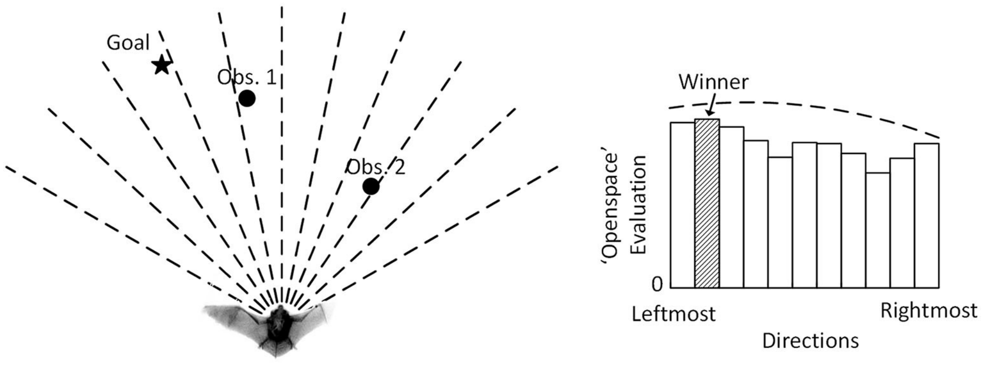


FIGURE 1 | The Openspace algorithm of Horiuchi (2009). (Left) An echolocating bat that is attempting to fly to the goal (filled star) while avoiding two obstacles (filled circle). (Right) The evaluation pattern consists of a constant plus a wide low-amplitude Gaussian (Goal input) with two dips created by the suppression from the two obstacles. A winner-take-all (WTA) function selects the direction with the highest evaluation (filled bin). The dashed line indicates the default evaluation with no obstacle present.

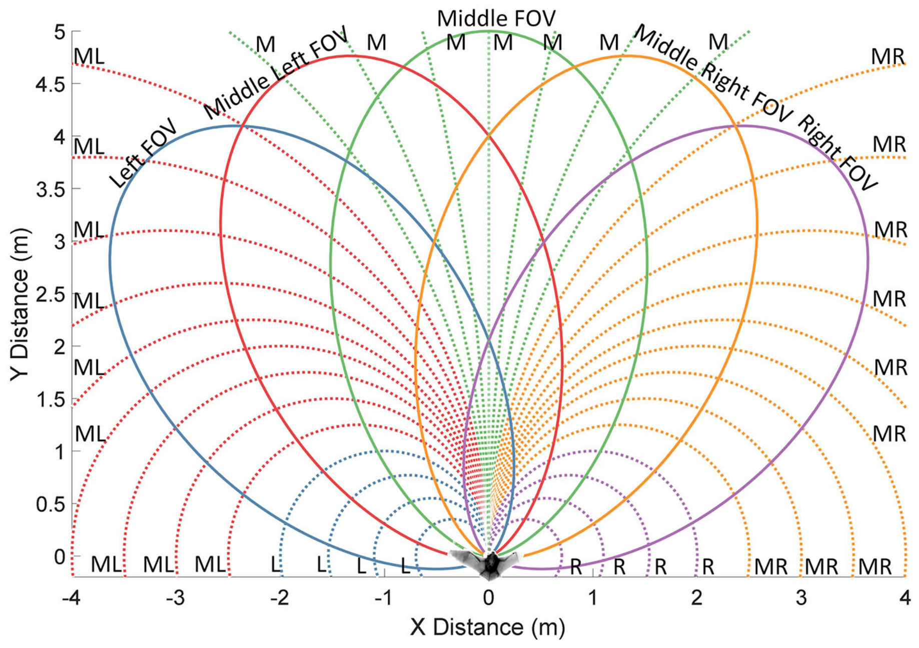


FIGURE 2 | An illustration of motor choices, sonar field of view (FOV) and groupings of the motor choices. Selecting different motor choices results in paths as circular arcs with different radii (dotted lines). The sonar FOV (solid lines) is ellipsoidal and the bat can turn its head to ping in different directions. Five ping directions and 33 motor choices are shown. The colors along with the letters at the end of the paths indicate which groups (left, middle-left, middle, middle-right or right) the motor choices belong to.

ping. A practical sonar system, however, has a limited field-of-view whose detection range is angle-dependent due to the beam patterns of both the sonar transmitter and the receivers, resulting in an ellipsoidal FOV (**Figure 2**). For big brown bats, the half-power beamwidth (the angular width of the beam pattern at the 3 dB cutoff points) of their emitted ultrasonic signal at 35–40 kHz is ∼56–80 degrees (Ghose and Moss, 2003; Gaudette et al., 2014). In the simulations shown here, a Gaussian-shaped FOV with a standard deviation of 30 degrees was used, resulting in a half-maximum width of 70.7 degrees. This limited FOV means that a bat will need to sequentially probe different directions to make good steering choices, which is time-consuming and leads to choices based on old data. Although engineered systems frequently employ continuous side-to-side scanning, this is *not* observed in echolocating bats flying through a field of obstacles. For a well-defined task like steering toward a goal, when a clear path toward the goal is detected, no scanning is needed. In this paper, we propose a novel neural model to find a collision-free

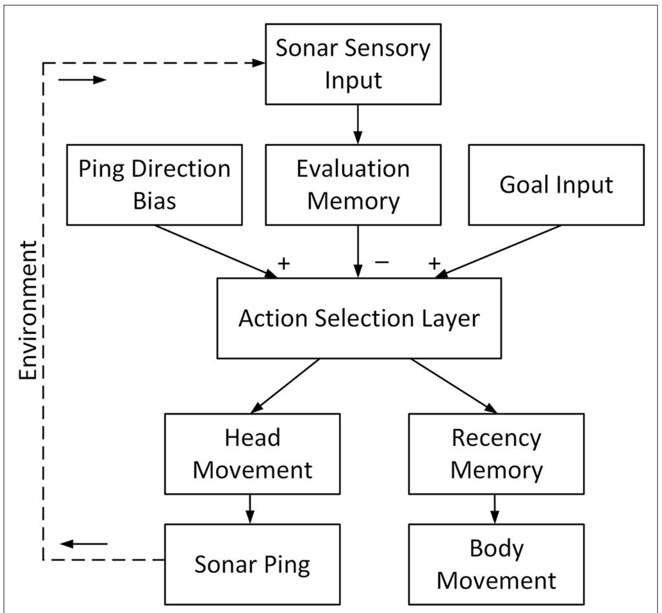


FIGURE 3 | The system structure of the proposed model. After a sonar ping, the sensory input updates the evaluation memory. The evaluation memory, representing the risk of each path, inhibits the action selection layer. The direction of the current ping excites the action selection layer to encourage selecting paths with the most recent information. The goal input represents the direction of the goal and excites the action selection layer. The action selection layer then selects the winning path with a winner-take-all (WTA) function and the winning path drives the head movement of the bat to direct its next ping in the associated direction. In the meantime, the recency memory checks if the sensory information on the winning path is recent. If so, the winning path drives the bat to turn its body and fly on it.

path using attentional search to guide the movement of sensors. The selected paths correspond to the natural curvature of bat flight.

The Curved Openspace neural model, like the original Openspace algorithm, combines echoes from a 2-D (azimuth and range) sonar to create an evaluation for each of the different motor actions under consideration. Due to the limited field-of-view of the sonar, a memory is needed to hold these evaluation values as the sonar interrogates different directions. To avoid excessive scanning, we introduce an attentional system that integrates information about a goal direction and stored action evaluations to determine where (or if) to turn the sonar for the next ping. Instead of constructing and updating an expensive 2-D map about all the obstacles in the memory of a bat, we can build a significantly simpler system by collapsing the 2-D sensory map into a 1-D evaluation memory among different motor action choices. Each motor action represents an arc of travel through the environment and its evaluation represents how risky it is. Having the evaluation memory to combine sensory inputs across head turns enables an action selection layer to select the most desirable path using a winner-take-all (WTA) function. Unlike the immediate motor response of the original Openspace algorithm with each sonar ping, the action selection (WTA) layer only provides preliminary decisions that are further processed before making a final decision. It should be noted that although the proposed model is described with a sonar system, it can work with any type of input sensor without significant changes to the structure of the model.

We describe the neural model in detail in Section The Curved Openspace Neural Model and we propose a spiking neural model in section spike-latency neural model that takes advantage of the inherent time representation that originates in the sonar. In section experiment results, we show the simulation results of the proposed model in a dense forest where we collect statistics and investigate the effects of different features and parameters, and we show the simulation of the spiking neural model with comparable performance. We also validated the use of the model on an inexpensive mobile robot with a limited-FOV sonar in a "pipe forest."

THE CURVED OPENSPACE NEURAL MODEL

The purpose of the Curved Openspace model is to generate a series of motor actions driven by a realistic sonar system to guide a bat-like agent to a goal location while avoiding obstacles on a 2-D plane.

The bat first localizes obstacles inside its field of view and sends the information to the evaluation memory (**Figure 3**) via a mapping that reflects different motor actions. The evaluation memory then combines new information with previous results to provide an evaluation of the *risk* along the trajectories of different motor choices. It suppresses the action selection layer with a Gaussian-shaped projection of inhibition. The action selection layer is excited by the goal input and uses a WTA function to calculate the most desirable motor choice. To fine-tune the action selection, other inputs, such as winner hysteresis and ping-direction bias, are added. Following the WTA, a "recency" condition is used to decide if the selection of the winning motor action (i.e., path) was based on recent information. If so, the bat is confident that the path is clear and executes the winning motor action. If the sonar has not sampled the winning motor action's direction recently, the bat will then ping in that direction, but will not execute the winning motor choice until it has.

In this model, it is assumed that the bat has a limited selection of motor choices, which includes a straight path and several curved paths (**Figure 2**). The straight path represents moving straight forward, and the curved paths are circular arcs with different radii consistent with the bat flying with a fixed turning rate. The bat is limited to ping in five fixed directions with respect to the body orientation (**Figure 2**). Depending on the portion of a path that falls into the field of view of a ping direction, individual paths are assigned to one of five view groups so that each path is associated with a single ping direction (**Figure 2**). We will refer to this neural model as the "analog" model when comparing it to a different implementation in Section Spike-Latency Neural Model.

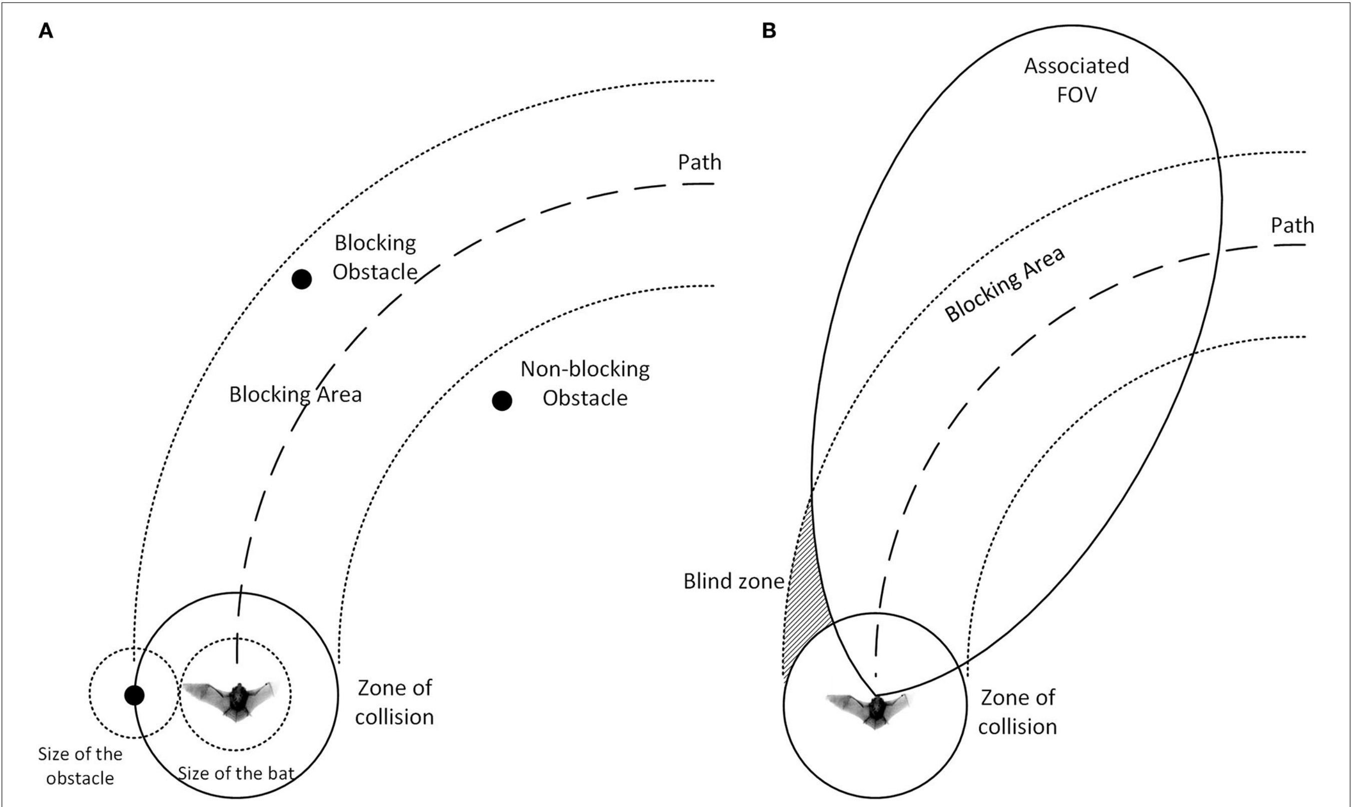


FIGURE 4 | (A) The definition of the zone of collision. The zone of collision is a disk around the bat with a radius equal to the sum of the radius of the bat and the radius of the obstacle. With this definition, we can treat obstacles as points and the bat as a disk. The area where the obstacles are blocking the path (dashed line) is defined as the blocking area (dotted line). **(B)** An example of the "blind zone" where the associated FOV cannot cover. Obstacles can disappear from the FOV and move into the blind zone while still blocking the path. Taking the maximum value between the memory and the most recent evaluation can help the bat remember the existence of the obstacle and steer away accordingly.

Zone of Collision

We define the radius of the bat to be its maximum wingspan and we define the zone of collision to be a disk around the center of the bat (**Figure 4A**). The bat is considered to collide with an obstacle if the obstacle passes into the zone of collision. If the minimum distance between an obstacle and a path is smaller than the radius of the zone of collision, the obstacle is defined to be a blocking obstacle since traveling on this path will eventually lead to a collision. In this paper, the bat is modeled without flapping wings and cannot perform agile maneuvers with wings to avoid contact with obstacles as might occur in the real world.

Speed Control

The traveling speed of the bat is controlled according to a "speed profile" that assigns a certain constant speed to each trajectory. The speed along a path p is selected as

$$v_p = V_{\text{MAX}} \cdot \gamma_p \tag{1}$$

where V_{MAX} is a constant maximum speed and γ is the speed profile that varies from 0.0 to 1.0 for each path. Because the sonar is only able to observe different distances along each trajectory, the speed is adapted to normalize the risk for comparison across a fixed amount of time (i.e., the time between sonar pings). The straight, middle path has the longest observability and thus supports the highest speed. Hence, the speed profile can be calculated as

$$\gamma_p = R_p / R_{\text{MAX}} \tag{2}$$

where R_p is the length of path p in its associated FOV and R_{MAX} is the maximum distance that the sonar system can detect. The speed profile aims to give the bat a constant reaction time on each path between a blocking obstacle becoming detectable and colliding with the bat. Intuitively, if the bat can detect obstacles further into the future on a path, it should be more comfortable with flying at a higher speed on that path. Sharp turns, however, usually require the bat to fly at a lower speed since the FOVs to the side only cover a short length of the path and any detectable obstacles in the blocking area are already close to the bat (**Figure 2**). Implementing the "speed profile" for speed control reduces the chance of collision during sharp turns.

Evaluation Memory

The 1-D evaluation memory represents the collision risk of different paths based on new and old sensory information. Each bin of the evaluation memory corresponds to a motor action choice. The analog value of the memory represents "risk" that combines the immediacy of avoidance and the concept of the growing uncertainty of the locations of blocking obstacles due to the movement of the bat. The *immediacy* of avoiding a blocking obstacle j on a path p is represented as a function (f).

$$f_p\left(j\right) = E_{\text{MAX}} \cdot \left[1 - \frac{r_p\left(j\right) - R_C}{R_{\text{MAX}} \cdot \gamma_p}\right] \tag{3}$$

where E_{MAX} is a constant representing the maximum value of f_p. $r_p\left(j\right)$ is the distance from the bat to a blocking obstacle j along path p approximating the obstacle to be on the closest spot on path p. R_C is the radius of the zone of collision, and $r_p\left(j\right) - R_C$ represents the distance that the bat can travel along path p before colliding with obstacle j. R_{MAX} is the maximum distance that the sonar system can detect and γ_p is the speed profile for path p. According to Equation 1 and the relationship that $r_p\left(j\right) \leq R_p$, the second term in Equation 3 is smaller than 1, resulting in a positive $f_p\left(j\right)$. The second term indicates how long the bat can travel on path p before colliding with a blocking obstacle j. A closer blocking obstacle on a path with a higher speed profile will result in a higher immediacy f_p.

To evaluate the risk of a given motor action choice p (a path), we use the following equation:

$$\underbrace{\text{E}_p(t)}_{\text{New value}} = \max\Big[\underbrace{E_p(t-\Delta t)}_{\text{Previous value}} - \underbrace{\beta(t-\Delta t)\cdot I_d}_{\text{Passive decay}} - \underbrace{\alpha(p)\cdot I_{inh}}_{\text{Directional inhibition}},\ \underbrace{min\left(\sum_{(j\epsilon B_p)} f_p(j), E_{\text{MAX}}\right)}_{\text{Summation of the immediacy of avoidance (saturating)}}\Big] \tag{4}$$

where

$$\alpha(p) = \begin{cases} 1, & \textit{if path p is associated with the direction of the current ping} \\ 0, & \textit{otherwise} \end{cases} \tag{5}$$

$$\beta(t) = \begin{cases} 0, & \textit{decay is halted} \\ 1, & \textit{otherwise} \end{cases}$$

where $E_p(t)$ is the evaluated risk for a path p, and the new risk is the maximum between a decayed memory and a saturating summation of the immediacy of avoiding blocking obstacles. $\beta(t)$ represents a passive decay whose strength I_d is a constant. The passive decay represents the increasing uncertainty of the location of obstacles with time and is updated at every time step except when the bat enters a "scanning mode" and halts the passive decay. $\alpha(p)$ is a binary value that becomes 1 when the bat sends a ping and path p is associated with the direction of the ping. It represents a directional inhibition where the strength of the inhibition I_{inh} is a constant. B_p is a set of the obstacles blocking path p and it defaults to an empty set when the bat does not send out a sonar ping at time t, resulting in a summation term of 0. This summation term represents "risk," where obstacles with higher immediacy of avoidance pose a higher risk and this risk accumulates with every blocking obstacle along the path. The risk is then saturated (using the minimum function) if it gets higher than the maximum evaluation value E_{MAX}. The final evaluation value after the update is the larger of the decayed old value and the new risk value computed from the objects currently being sensed. Because the FOV that a path is associated with cannot cover the whole blocking area of the path, a blocking obstacle can disappear from the current field of view while still being a threat (**Figure 4B**). The max function is thus used as a more conservative assessment of risk between the memory and what the sensory system is detecting. Essentially, the evaluation memory is updated with new information if a sonar ping is sent at time t, and the memory is kept with or without decay (depending on β) when there is no sonar ping.

The evaluation memory allows the bat to combine path evaluations gathered through several pings in different directions. Since the bat is still moving while gathering information, the stored memories can become outdated. By default, the values of the evaluation memory decay quickly over time to represent an increase in the ambiguity of obstacles (constant "passive decay"). When the bat changes its ping direction, the decay is temporarily halted until a motor choice has been selected and executed. In this "attentional search mode," the bat pings rapidly in the different directions of interest to minimize the distance traveled between pings and the loss of accuracy in the memory due to movement.

When a sonar ping occurs, the evaluation memories of the paths associated with the current ping direction are also inhibited. This "directional inhibition" is shown as the term $\alpha\left(p\right) \cdot I_{inh}$ in Equation 4. Since the maximum function will keep the previous evaluation memory if the bat did not detect any obstacles blocking the path, the path might be more open than what the memory suggests. The directional inhibition aims to reduce the risk of a path in this scenario.

Action Selection Layer

In this model, collision avoidance is viewed as an attentional search for good paths, combining parallel search within the field of view of a single ping, but serial search across head movements (Itti and Koch, 2000). The action selection layer combines the collision risk calculation from the evaluation memory with goal information and other biasing signals to create a desirability value for each path (i.e., motor) choice. Having no obstacle on a path will give that path a low risk value, resulting in a high desirability in the WTA layer. Calculation of the desirability of each path occurs after every sonar ping and can be described by

$$D_p = \underbrace{D_0\left(p\right)}_{\text{Constant Bias}} + \underbrace{G \cdot e^{\frac{-\left(p-p_g\right)^2}{\sigma_g^2}}}_{\text{Goal input}} + \underbrace{P \cdot \alpha(p)}_{\text{Ping Direction Bias}} + \underbrace{H \cdot e^{\frac{-\left(p-p_h\right)^2}{\sigma_h^2}}}_{\text{Hysteresis}} \underbrace{- W \cdot \sum_{p_{mem}=1}^{N} E_{p_{mem}} \cdot e^{\frac{-\left(p-p_{mem}\right)^2}{\sigma_m^2}}}_{\text{Inhibition from evaluation memory}} \tag{6}$$

The first term $D_0(p)$ is a positive constant bias that represents default desirability. This term allows the evaluation to remain positive even while being reduced by other terms. Besides, each path could have a different bias term to incorporate additional information about the desirability of individual paths due to actuation limits or energy considerations. This was used in the simulations described in section experiment results to discourage sharp left or right turns. The coefficient G is the amplitude of an additive Gaussian term with its center at p_g and a standard deviation of σ_g. This term provides a bias toward some motor actions over others due to externally provided information about a goal location. The magnitude of excitation from the goal input is much weaker in comparison to the inhibition from the evaluation memory since the goal input only serves as a bias toward the goal when different motor actions have similar risks levels. The third term is an excitation term from the current ping direction that biases the bat to select the path with the most recent information when several paths have similar risks. $\alpha(p)$ is the binary value defined in Equation 5 and P is the amplitude of the excitation. The coefficient H is the amplitude of a Gaussian term that produces hysteretic behavior, where p_h is the path that the bat is currently traveling on and σ_h is a constant that controls the width of the Gaussian. This term prevents the bat from changing paths erratically due to noise in the measurements or occupancy calculations. The index p_{mem} refers to the bins of the evaluation memory that suppress the desirability with a subtractive Gaussian term scaled with their values $E_{p_{mem}}$ and an inhibition weight of W. The width of the suppression is controlled by σ_m that is kept constant. After the evaluation, the action selection layer selects the path with the maximum desirability for head and body control.

Motor Control

The action selection layer determines the direction in which the head should be pointed (i.e., in the direction of the winning path). If a head movement is needed, it will be performed. There are two scenarios in which a path is selected. The first scenario is when the path has a low risk value after the bat has pinged in its direction. In this case, the bat will likely travel along this path. The second scenario is that the bat has not pinged in the direction of the winning path, and its desirability is high because the default risk value in the evaluation memory is low. In this situation, pinging in an unknown direction can help the bat explore possible open paths.

For the bat to fly along a particular trajectory (i.e., *execute* a motor action,) the path must be chosen by the action selection layer ***and*** the sonar data evaluating that path must be fresh (a.k.a., "recent"). If the selected path is based on old data, a head movement (and a ping) is generated to obtain new data. Once both criteria are satisfied, the output of the action selection layer is allowed to change the bat's trajectory. At the same time, the bat exits "attentional search mode" and begins allowing its evaluation memory to decay. To keep track of data recency, each of the five ping directions has a countdown timer called recency memory that resets to a high value after a ping in its assigned direction. The recency memory of a ping direction must exceed a certain threshold for the bat to select paths associated with that direction. Note that the bat continues to travel along its prior trajectory until a new path (motor action) is selected for execution. An example of the behavior of the proposed model is shown in **Figure 5**.

When the desirability of the winning path is lower than a certain threshold (i.e., no acceptable paths were detected), an "emergency" is declared and the bat will do a sharp 180-degree turn and travel in the opposite direction to the path it was traveling on before the turn. It will also direct its next sonar ping to the associated direction of the traveling path. This emergency "turnaround" maneuver is vital in the simulation for the bat to escape from the scenario where all paths are blocked by obstacles (a.k.a., a trap). Bats in the real world often perform this by using the vertical dimension to abruptly fly straight up, turn around and fly back down in the opposite direction (similar to the "hammerhead" maneuver in airplanes).

The Curved Openspace model was simulated in a dense forest where we investigate the effects of different parameters in section simulation of the analog model. The model was also implemented on a mobile robot with a car-like steering mechanism in section robot implementation. We show that the robot is able to travel in a dense forest of plastic pipes without collision using a narrow-FOV sonar system mounted on a head-turning servo motor.

SPIKE-LATENCY NEURAL MODEL

Although the analog model presented above can be implemented using large populations of spiking neurons to simulate (noisy) analog signal representations, spike-timing-based signal representations often suggest very different neural implementations with far fewer neurons. In this section, a spiking neural network model using spike-timing to represent signal values is described. As shown in **Figure 6**, its structure consists of four main layers: a sensory layer that encodes the 2-D locations of obstacles, a memory layer that integrates and stores the sensory information associated with different paths ("Evaluation Memory"), an "Action Selection" layer that uses a "race-to-first-spike" winner-take-all (WTA) mechanism to select a path, and a "Motor" layer that implements the body steering decision and head movements. Inputs to the spiking neural model also include ping directions (head direction), a global reset signal, goal direction input, and a ping onset signal.

The time at which a neuron fires a spike following an outgoing echolocation pulse is affected by many variables. Echolocation is foundationally based on the time-of-flight of sound to determine the distance to objects, with the closest objects generating echo signals first, producing a natural temporal coding scheme. Additionally, echoes from a given object are louder if it is closer. Interestingly, neurons commonly exhibit shorter latency responses to larger magnitude signals. In this model, following each sonar ping, sensory neurons will fire spikes with latencies that reflect the immediacy of avoiding any detected obstacles. The evaluation memory units integrate the sensory information on different paths and store the integrated information as spike latencies using a delay line and an array of latency memory units, which will be described in detail in Section Evaluation

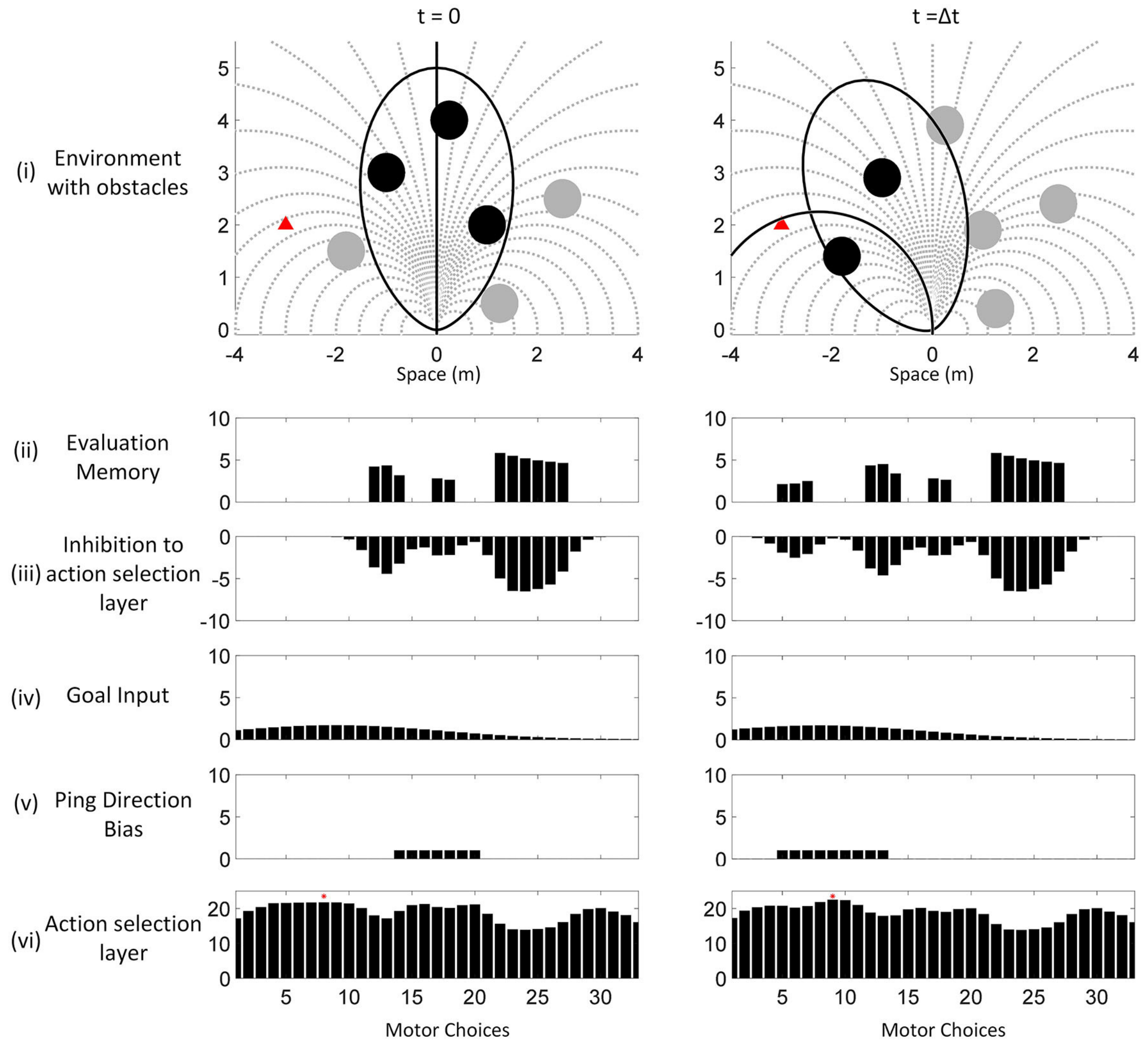


FIGURE 5 | An example of the proposed neural model from sensory input to motor action selection. The environment, evaluation memory, values of the action selection layer, ping direction and motor actions in two timesteps from the simulation are shown. A bat was flying in a field with 6 obstacles (filled circle) and a desired destination (filled triangle in red). The circles of the obstacles are drawn with the size of the zone of collision to show the paths that they are blocking. The bat has a limited selection of motor choices (dotted line) and each of the motor actions corresponds to a bin in the bar graphs below. At $t = 0$, it is assumed that the bat arrived in the environment (i) with no prior information. It is also assumed that it was flying along a straight path (solid straight line) and it directed its first sonar ping to the front (ellipsoidal FOV shown in solid line). With the first ping at $t = 0$, it detected three obstacles (filled circle in black) and updated its evaluation memory (ii). The evaluation memory units suppressed the action selection layer with Gaussian projection of inhibition (iii) while the goal input imposed a wide Gaussian excitation (iv). Because the bat sent out its sonar ping toward the front, the associated motor actions received excitations as shown in the ping direction bias (v). The action selection layer (vi) selected the winning path (indicated with a red dot on top of the bar), which is gated by the recency memory (not shown). Because the winning path is associated with the middle-left ping direction and the bat had not pinged in that direction recently, it turned its head to send the next sonar ping to the middle-left, did not execute the winning path and entered the "scanning mode" that halted the passive decay on the evaluation memory. At the next timestep $t = \Delta t$, the bat pinged to the middle left direction and updated the evaluation memory. The goal bias stayed the same while the ping direction bias changed to excite the paths associated with the middle left direction. The action selection layer combined the information in the same way and selected a path as the winner. Since the winning path belongs to the middle-left group and the bat just pinged in the same direction, the bat executed the winning path (solid line), exited the "scanning mode" and allowed the evaluation memory to decay. Notice that with the proposed model, the bat only pinged in the directions of interest to find the most desirable path and did not need to do a full scan.

Memory. Because the stored signals are spike latencies, during readout, the evaluation memory units must be able to re-generate output spikes with the same latencies without sensory input. The evaluation memory sends out spikes with the stored latencies to the action selection layer, where the net desirability of different paths is compared, and a "winner" is selected. The neuron that fires the first spike in the action selection layer has the highest desirability and inhibits all other neurons in the same layer to prevent them from firing. The spike from the winning neuron is then sent to the motor neurons that will orient the sonar head for

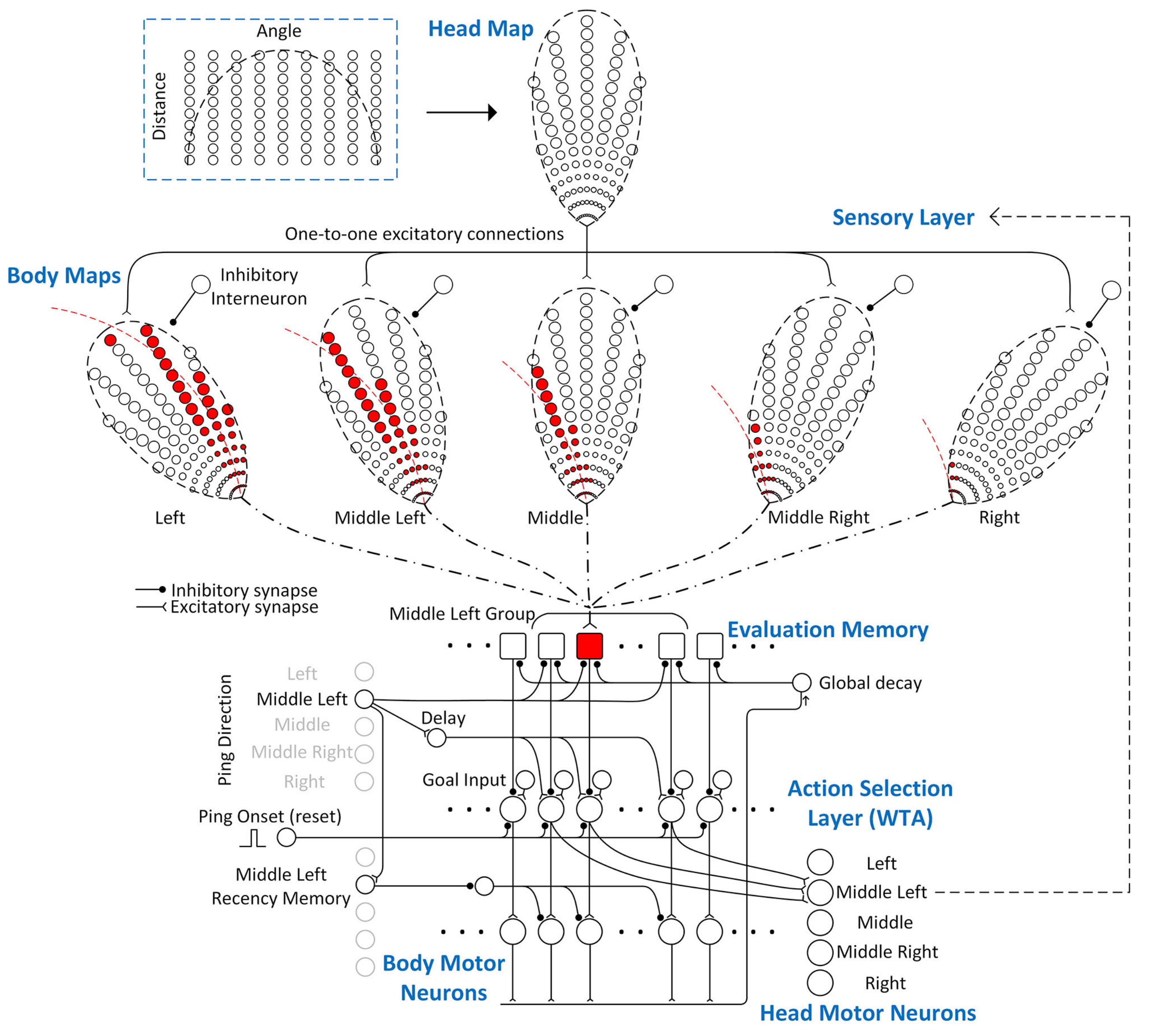


FIGURE 6 | The spiking neural model of the Curved Openspace model. It mainly consists of four layers: a 2-D sensory layer encoding the locations of obstacles, a memory layer storing the "risk" of different paths, an action selection layer that uses a "race-to-first-spike" WTA mechanism to select the winner and a motor layer that controls the body and head movements. Only a portion of the connections in the group of middle-left ping direction are shown in this example for clarity. The sensory layer consists of a 2-D sensory map in head frame (head map) and five sensory maps in body frame (body maps). The elliptical dashed lines around the maps represent the field of view of the bat and the circles represent sensory neurons which fire when obstacles are detected in their locations. The number of neurons in each map is reduced for clearer illustration. Each body map has an inhibitory neuron that inhibits all the neurons in the map when it fires a spike. The neurons in the head map strongly excite the neurons in the same positions in each body map but only one of the body maps will not be inhibited by the inhibitory interneuron after a sonar ping, depending on the ping direction. The neurons in the body maps make fixed excitatory connections (dash-dotted line) to the evaluation memory units if the represented obstacles in their positions are blocking the paths. One example of the connections between the body maps and an evaluation memory unit is shown in red.

the appropriate ping direction (head motor neurons). The spike is also sent to the body motor neurons that produce the turn rate needed for the corresponding path. The body motor neurons, however, will only be activated if the recency memory of the associated ping direction is active. The gating from the recency memory is implemented with a disinhibition mechanism. The following sections describe each layer in detail.

Sensory Layer

In this neural implementation, the sensory layer (**Figure 6**) uses a 2-D head map to represent the locations of obstacles and converts the information from the head reference frame to the body reference frame with several body maps. The neurons in the head map make strong one-to-one excitatory connections to the neurons with the same positions in all the body maps. The number of body maps is the same as the number of possible ping (i.e., head) directions and there is an inhibitory interneuron for each body map that strongly inhibits all the neurons in the map when it fires a spike. In this model, we define the spike latency as the time between the outgoing sonar ping and the first spike from a neuron. When the bat pings and detects an obstacle, the neuron at the corresponding location in the head map fires a spike with a latency close to the latency of the echo. Depending on the direction of the sonar ping, only one body map will be selected as active by inhibiting all other body maps using the inhibitory interneurons. A spike from the head map excites the corresponding neuron in the active body map and causes it to fire a spike immediately. The spikes from the active body map represent obstacles at certain locations in the bat's body frame. Because the Curved Openspace model uses the distance along a curved path (not the radial distance which is represented by the echo delay) to evaluate the risk value (Equation 3), the latencies of the spikes from the active body map are adjusted by adding extra latencies before the spikes are sent to the evaluation memory. Whether or not an evaluation memory unit takes synaptic inputs from neurons in the body maps is determined by whether obstacles in their positions are blocking the path that the evaluation memory unit represents. If they are blocking the path, the spikes from the corresponding neurons will be sent to the integrating neuron in the evaluation memory, which will be described in detail in section evaluation memory. The synaptic connections between the sensory layer and the evaluation memory are fixed if the paths are fixed.

The Openspace algorithm in Horiuchi (2009) made clever use of the natural latency of echoes as a representation of the straight-line (i.e., radial) distance to obstacles. In the Curved Openspace model, however, the immediacy of avoidance uses the distance along a curved path (Equation 3). As described earlier, to adjust the spike latency to reflect the immediacy correctly, each neuron in the body map connects to a delay neuron (not shown in **Figure 6**) that adds a *constant* latency specific to each position before connecting to the evaluation memory. The added latency for a connection between a sensory neuron j to the evaluation memory of path p can be calculated as

$$\Delta t\left(j,p\right) = T_{\mathrm{MAX}} \cdot \frac{r_p\left(j\right) - R_C}{R_{\mathrm{MAX}} \cdot \gamma_p} - t_{echo}\left(j\right) + T_C \tag{7}$$

where $\Delta t\left(j,p\right)$ is the added latency and $t_{echo}\left(j\right)$ is the echo latency from an obstacle represented by neuron j. $t_{echo}\left(j\right)$ is constant for each sensory neuron since each neuron represents a fixed location on the body map. T_{MAX} is the echo delay from an obstacle at the maximum sensing distance R_{MAX} and is the maximum echo delay the sonar system can receive. The term $\frac{(r_p(j)-R_C)}{(R_{\mathrm{MAX}}\cdot\gamma_p)}$ is the same term used in the immediacy function (Equation 3) that represents the time before the bat collides with an obstacle at the location of neuron j. $r_p\left(j\right)$ is the distance from the bat to a blocking obstacle j along path p, R_C is the radius of the zone of collision, R_{MAX} is the maximum distance that the sonar system can detect and γ_p is the speed profile for path p. The term has a value from 0 to 1, which makes the first term in Equation 7 have a value between 0 and T_{MAX}. T_C is a small and constant delay to keep $\Delta t\left(j,p\right)$ positive. With the added latency, the spike latency that arrives at the evaluation memory of path p from sensory neuron j (if an obstacle is sensed) can be written as

$$t_{\mathrm{spk}}\left(j,p\right) = \Delta t\left(j,p\right) + t_{\mathrm{echo}}\left(j\right) = T_{\mathrm{MAX}} \cdot \frac{r_p\left(j\right) - R_C}{R_{\mathrm{MAX}} \cdot \gamma_p} + T_C \tag{8}$$

A closer blocking obstacle on a path with a higher speed profile will result in a shorter spike latency, indicating a higher immediacy of avoidance.

Evaluation Memory

The role of the evaluation memory (as described in Section Evaluation Memory) is to hold the spatially integrated value of immediacy along each path even when the sonar is interrogating a different direction and does not receive new sonar information for a given path. Given the spike latency representation, the output of the evaluation memory unit is a spike with a latency (following the sonar ping) that matches the previously observed latency (when the sonar was receiving new data). Each evaluation memory unit receives spikes with different latencies from the sensory neurons along a path and integrates them into a spike latency with the integrating neuron. An array of memory units along with a delay line detects and stores the occurrence of a spike at a particular latency and is then able to regenerate the spike upon later activation. The neural circuit of each evaluation memory unit is shown in **Figure 7A**.

An example of the mechanism of the integrating neuron is shown in **Figure 7B**. The integrating neurons are integrate-and-fire neurons and their membrane potentials simultaneously reset when the bat pings ($t = 0$). Whenever the integrating neuron receives a spike, a step-excitation current is turned on (for 20 ms), causing its membrane potential to rise. If the membrane potential reaches a threshold, the integrating neuron fires a spike. The spike latency is shorter when excitatory spikes arrive earlier (representing stronger inputs). The excitatory currents from different spikes are summated and can further reduce the latency of the spike.

For an integrate-and-fire neuron with a membrane capacitance C_{mem0}, a spike threshold V_{th0}, and n (*where* $n \geq 1$) step-excitation currents each with an amplitude of E_0 activated at time $t_1, t_2, \ldots, t_n$, the latency of the spike T_{eval} is given by

$$T_{\mathrm{eval}} = \frac{C_{\mathrm{mem0}} V_{\mathrm{th0}} + E_0 \cdot \sum_{i=1}^{n} t_i}{n \cdot E_0} \tag{9}$$

assuming that $t_1 \leq t_2 \leq \cdots \leq t_n < T_{\mathrm{eval}}$. Each input spike that arrives before the output spike reduces the spike latency, but the latency cannot be reduced below the latency of the first input spike. The spike latency T_{eval} represents the integration of immediacy of avoidance with a saturation limit similar to the second term in Equation 4, although the integration is different from simple addition. If the integrating neuron receives one or more spikes from the sensory layer, it will fire a spike with a latency of T_{eval} which resembles the evaluated risk of a path. Different from the risk calculation in the analog model (Equation

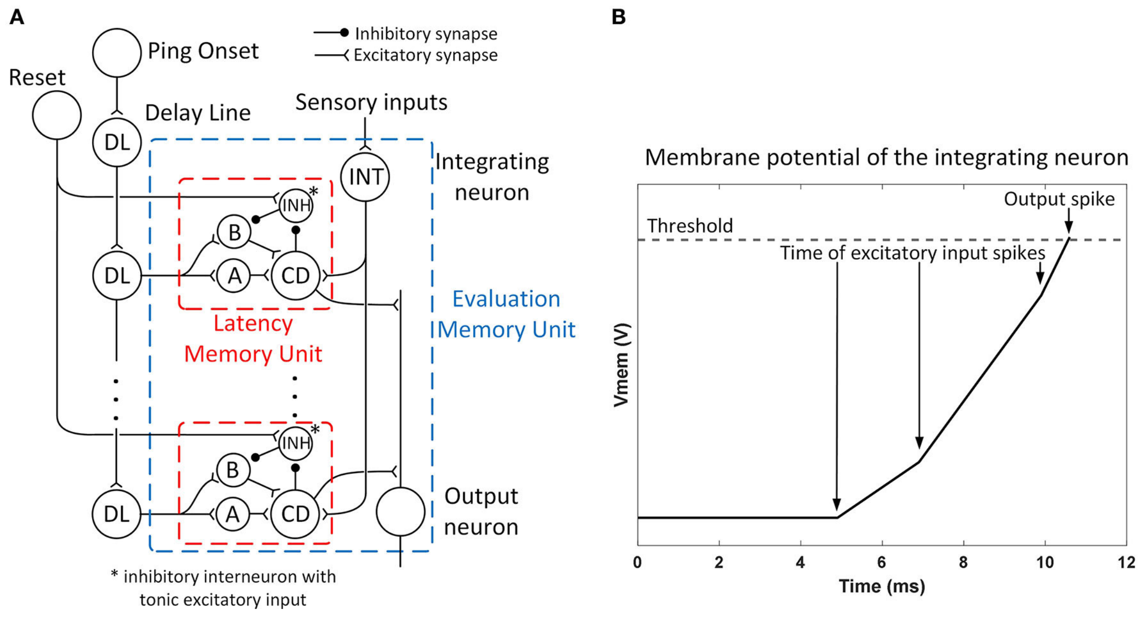


FIGURE 7 | (A) The structure of an evaluation memory unit. The integrating neuron combines the spike train from sensory neurons into a single spike with a latency representing the risk of the corresponding path. The spike latency is then stored in an array of latency memory units. The memory can be reset by exciting the reset neuron. The output neuron combines the spikes from latency memory units into a single spike train and sends it to the action selection layer. **(B)** The mechanism of the integrating neuron. Each input spike triggers a step-excitation current that charges up the membrane potential V_{mem} and the currents from different sensory neurons accumulate. A spike is fired when V_{mem} crosses a threshold. An input spike that arrives earlier will result in a shorter spike latency. Input spikes after the first one also reduces the spike latency by increasing the charge speed.

4), here a smaller latency means a larger risk value. The latency of the spike then needs to be stored in the evaluation memory.

An array of latency memory units and a delay line are used to store the latency of the spike from the integrating neuron and later generate a spike with the same latency when needed without the original sensory inputs. As is shown in **Figure 7A**, each evaluation memory unit has an array of latency memory units while the delay line is shared among all the evaluation memory units. The delay line is triggered by the onset of the sonar ping, and it generates spikes with different latencies which are sent to different latency memory units in different evaluation memory units. In this neural model, the delay line is implemented as neurons connected in series with excitatory synapses and each spike from the previous neuron causes the next neuron to fire with a fixed delay. The delays between neurons in the delay line affect the resolution of the latency memory.

As shown in **Figure 7A**, a latency memory unit consists of four neurons: two excitatory interneurons (A and B), a coincidence detector (CD), and an inhibitory interneuron (INH) with tonic excitatory input. Neurons A and B both take excitatory input from a neuron in the delay line, but neuron B also takes inhibitory input from neuron INH. Besides the tonic excitatory input, neuron INH is strongly inhibited by the CD neuron and strongly excited by a reset neuron. Without the input from the CD and the reset neuron, neuron INH fires tonically and keeps neuron B inhibited. The CD takes excitatory inputs from neuron A, neuron B, and the integrating neuron. For a CD to fire a spike, two spikes need to arrive at approximately the same time, meaning that two out of the three neurons exciting the CD need to fire simultaneously.

During the idle state before a sonar ping, neuron B is inhibited by neuron INH. When the bat pings, the ping onset starts the spike propagation in the delay line, and a spike with a certain latency will be sent to neurons A and B. Neuron A will be excited by the input spike and send a spike to the CD, whereas neuron B will not fire because it is strongly inhibited by neuron INH. At this point, only a spike from the integrating neuron with the same latency as the spike from the delay line will be able to trigger a spike from the CD. If this is the case, the CD will send a spike to neuron INH and keep it from firing again for the duration of the inhibitory synaptic input (around 300 ms). If the bat pings again while neuron INH is still inhibited, neurons A and B will both fire and the CD will fire again even without the spike from the integrating neuron. Since the CD fires again, neuron INH is kept inhibited for another interval, allowing the next sonar ping to cause neuron B to fire. Unless neuron INH is reset by the reset neuron or the bat doesn't ping for a long time, the memory of the spike latency from the integrating neuron is maintained and a spike with the same latency is reproduced after every sonar ping without any further sensory inputs.

The CD also sends an excitatory spike to the output neuron when it fires. The output neuron in each evaluation memory unit fires a spike with very little delay whenever it receives spikes from any of the latency memory units. Like an OR function, it combines all of the spikes from the delay-tuned neurons into a spike train. Due to the mechanism of the action selection layer,

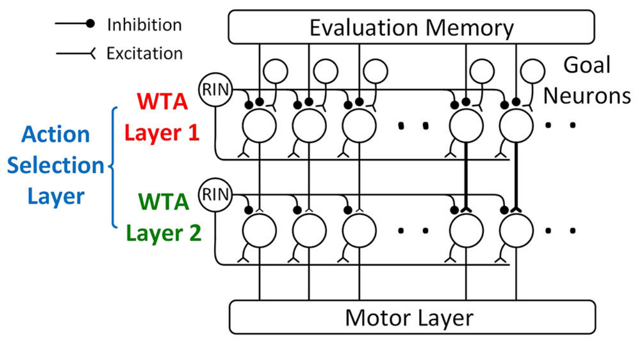


FIGURE 8 | The action selection layer is composed of two WTA layers: WTA layer 1 and WTA layer 2. Both layers use a "race-to-first-spike" WTA mechanism, where the first neuron to spike excites a recurrent inhibitory neuron (RIN) to fire that suppresses other neurons within the layer from firing. Because WTA layer 1 can produce multiple winners that represent significantly different turning, WTA layer 2 is added to only allow the winners from the same ping direction group to go to the motor layer. In WTA layer 2, neurons toward the center (straighter paths) have a stronger synaptic connection from WTA layer 1 (represented by thicker synapses in the figure), giving those paths priority over paths toward the side (sharper turns).

however, as will be described later in Section Motor Layer, only the latency of the first spike in the output spike train affects the calculation of the desirability. In the scenario where an evaluation memory unit with a stored spike latency receives new sensory input, only the spike with the shorter latency is meaningful to the next layer. Since a shorter spike latency represents a higher risk value, this behavior is consistent with the description earlier in Section Evaluation Memory that the final risk value is the maximum value between the decayed old risk and the new risk computed from the objects currently being sensed.

Action Selection Layer

The action selection layer consists of two WTA layers (**Figure 8**) with a similar structure, and both of the layers use a "race-to-first-spike" WTA mechanism to select the winning motor actions.

WTA layer 1 is similar to the temporal WTA circuit proposed in Horiuchi (2009). It compares the desirability of different paths using the latency of the spikes from the evaluation memory layer and selects the most desirable path with a "race-to-first-spike" WTA mechanism. WTA layer 1 consists of action selection neurons with an integrate-and-fire mechanism, a recurrent inhibitory interneuron (RIN), and a group of goal neurons (**Figure 8**). The numbers of action selection neurons and goal neurons are the same as the number of motor actions. The action selection neurons in WTA layer 1 receive weak excitatory input from the goal neurons that indicate the location of the goal. Each goal neuron connects to the field of action selection neurons with a Gaussian-shaped pattern of synaptic strengths (not all the connections are shown in **Figure 8** for clarity). Only one of the goal inputs will fire a spike to indicate the path that leads to the goal. This excitatory connection corresponds to the Gaussian-shaped goal input in Equation 5 described in section action selection layer. The action selection neurons also receive weak excitatory input from the corresponding ping direction neuron through a delay neuron ("Delay" in **Figure 6**). Upon receiving a spike from the ping direction neuron, the delay neuron fires a spike after some delay to the action selection neurons associated with the same ping direction. This excitatory connection corresponds to the ping direction bias term in Equation 6. In addition to the connections from goal and ping direction neurons, all the action selection neurons receive passive excitatory currents that reflect the baseline desirability of different paths, corresponding to the "Constant Bias" term in Equation 6. This excitatory current could be either from a neuron firing tonically or intrinsic membrane currents (Häusser et al., 2004).

When a sonar ping is emitted, the ping onset neuron simultaneously resets (i.e., strongly inhibit and then release) all of the action selection neurons. The passive excitatory currents can then be inversely expressed in the spike latency across the field of neurons (**Figure 9A**). In the absence of other inputs, the neuron that receives the strongest excitatory current will integrate to the threshold first and is the winner, meaning that the motor action with the largest constant bias will win. The excitatory spikes from the goal and ping direction neurons increase the membrane potential, thus decreasing the amount of charge that the IF neurons need to reach the firing threshold and making them more likely to win (**Figure 9B**).

Each evaluation memory unit is connected to all of the action selection neurons through inhibitory synapses that activate a long-lasting step-inhibition current if a spike arrives (not all connections are shown in **Figure 8** for clarity). The synapses have a Gaussian distribution of synaptic strengths with the peak centered on the synapse connecting the neurons representing the same path. Different synaptic weights mean different amplitudes of the activated inhibitory current. This way of connecting the evaluation memory and the action selection layer corresponds to the Gaussian inhibition term in Equation 5. The standard deviation of the Gaussian distribution, however, is small ($\sigma_c = 1$ in the simulation) and in practice synapses $3\sigma_c$ away from the center can be pruned without affecting the performance. The inhibitory current from a synapse saturates when the synapse receives a spike and any following spikes to the same synapse will not increase the amplitude of the inhibitory current. The saturating current from a synapse is the reason why only the latency of the first spike from an evaluation memory unit affects the computation of desirability.

The accumulated inhibitory currents from different synapses slow the rate of charging of the action selection neuron and the time to spike will increase as inhibitory inputs start earlier (**Figure 9C**). Because the spike latency from the evaluation memory is inversely related to the risk of different paths, a path with a higher risk will result in an earlier activation of inhibitory currents and thus increase the spike latency of the action selection neurons. In combination with passive currents and excitatory spikes from the goal and ping direction neurons, the action selection neuron that fires the first spike indicates the most desirable path.

For an action selection neuron with a membrane capacitance C_{mem}, a spike threshold V_{th}, a passive excitatory current I_{exc},

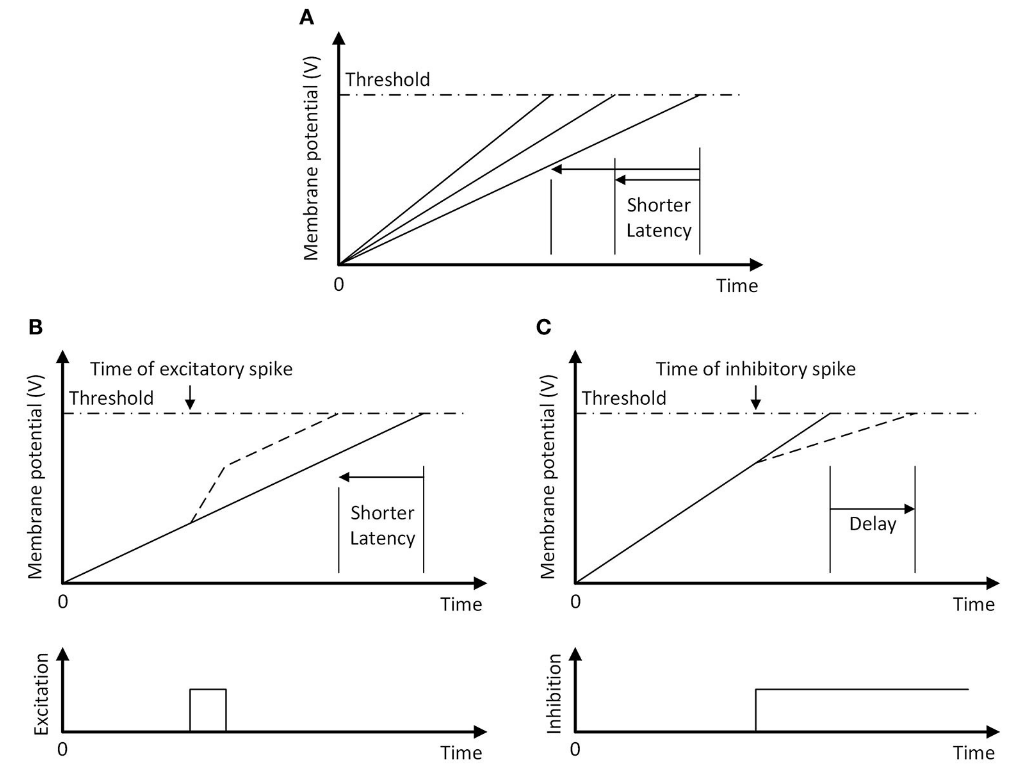


FIGURE 9 | The temporal WTA mechanism with an integrate-and-fire neuron. The neuron fires a spike when its membrane potential reaches the threshold (dash-dotted line). **(A)** Following a reset, increasing the passive excitatory currents increases the charging rate of the membrane potential and shortens the latency of the spike. **(B)** A pulse of excitatory current increases the membrane potential, shortening the spike latency (dashed line). **(C)** A long-lasting step-inhibition current increases the latency of the spike (dashed line) or prevents firing altogether. An inhibitory spike that arrives earlier produces a longer delay in firing.

a sum of injected charge from goal neurons Q_{goal} and ping direction neurons Q_{dir}, and n step-inhibitory currents activated at time $t_1, t_2, \ldots, t_{\text{n}}$, the latency of the spike T_{WTA} is given by

$$T_{\text{WTA}} = \frac{C_{\text{mem}} V_{\text{th}} - Q_{\text{goal}} - Q_{\text{dir}} - \sum_{i=1}^{n} I_i \cdot t_i}{I_{\text{exc}} - \sum_{i=1}^{n} I_i} \tag{10}$$

where I_i is the amplitude of the inhibitory current activated at time t_i. It is assumed that $t_1 \leq t_2 \leq \cdots \leq t_n < T_{\text{WTA}}$, meaning the spikes that arrive after the neuron fires are ignored. It is also assumed that $I_{exc} - \sum_{i=1}^{n} I_i > 0$, or else no spike is generated. As is shown in Equation 10, a larger passive excitatory current decreases the spike latency by increasing the denominator while stronger excitatory inputs from goal and ping direction neurons decrease the latency by reducing the numerator. Earlier arrival of inhibitory spikes (smaller t_i) increases the latency by increasing the numerator. Although the amplitudes of the inhibitory currents I_i affect both the denominator and the numerator, stronger inhibitions that arrive at the same time still increase the spike latency. In a cluttered environment where obstacles are blocking many paths, the sum of the inhibitory currents could exceed the sum of excitatory currents and prevent the neuron from firing a spike at all.

The first spike from the action selection neurons then causes a recurrent inhibitory neuron (RIN) to fire which recurrently inhibits every action selection neuron and prevents them from firing. Due to the non-zero latency of the RIN and the slow activation of inhibitory synapses, there is a delay between the first spike from the action selection neuron and the start of the recurrent inhibition. Because the passive excitatory current is weak and the excitatory inputs are short, the time it takes for the action selection neuron to fire a spike is much longer than the delay of the recurrent inhibition. As a result, each of the action selection neurons either fires a single spike or produces no spike at all. However, any spikes that occur before the inhibition is effective will pass through the WTA mechanism and multiple winners can appear at the output. This behavior is similar to the behavior of a k-WTA network which selects the k largest values. Although k-WTA networks have been shown to be useful in robot systems (Peng et al., 2021; Qi et al., 2021), having multiple winners at the output of the WTA layer in our proposed neural network could cause problems in both the head motor layer and the body motor layer. In our model of the motor layers, we assume that multiple activations of the motor neurons would cause averaging of motor actions. While multiple winners that encode nearby head directions and motor actions do not necessarily cause problems, winners representing significantly different paths will. To solve the problem with multiple winners, a second WTA layer ("WTA Layer 2" in **Figure 8**) is added to ensure that the winners sent to the motor layer are within the same ping direction group by eliminating some of the winners.

The structure of WTA layer 2 is the same as WTA layer 1 except for the input synapses. The action selection neurons in WTA layer 2 take one-to-one excitatory connections from the neurons in WTA layer 1. All of these connections are strong enough to fire the excitatory neurons with a single spike, and stronger synaptic connections produce spikes with shorter delays. The synaptic strengths are the same for neurons within each of the 5 ping direction groups but differ from group to group to create 5 different delays. The values of the delays are designed so that the minimal difference between delays is longer than the delay of the recurrent inhibition. As a result, although WTA layer 2 can still produce multiple winners, it is guaranteed that the winners are from the same ping direction group. The values of the synaptic strengths decide the priority of each ping direction group. It is beneficial in terms of safety and energy cost to favor slow turns over sharp turns, so in this model, the synaptic strengths in the middle group are set to be the highest, followed by the middle-left group, the middle-right group, the left group and then the right group.

Motor Layer

The function of the motor layer is to produce the motor actions needed to orient the head of the bat (head motor layer) or to turn the body (body motor layer) to follow a selected path.

Five head motor neurons representing five head (i.e., ping) directions receive excitatory input from the neurons in the action selection layer that are associated with the same ping direction (**Figure 6**). Upon receiving a spike, a head motor neuron becomes active and orients the head of the bat in the associated ping direction for the next sonar ping, consistent with the proposed attentional system described in Section Motor Control. Because only the neurons in the same ping direction group can win in the action selection layer, as described in Section Action Selection Layer, no more than one head motor neuron will be active at the same time.

Each body motor neuron is associated with a path, and when it fires, it produces the correct body/wing changes to execute the turn rate needed for its associated path. When multiple body neurons are active, the bat is modeled to produce the averaged turn rate and execute the averaged path, similar to the population coding hypothesis seen in other animals (Lee et al., 1988). The body motor neuron receives excitatory input from the action selection neuron associated with the same path and inhibitory input from an interneuron inhibited by the recency memory. Each recency memory is associated with a ping direction, and it is a decaying memory that stores the recency of its associated ping direction. The recency memory becomes active when a ping direction neuron fires, and it inhibits the interneuron for some duration until decayed back to its inactive state. The duration of the memory is decided by the duration of the excitation from the ping direction neurons. In this scenario, the inhibited interneuron can no longer inhibit body motor neurons and they are allowed to fire upon receiving spike input from the action selection layer. Through this disinhibition mechanism, a bat can turn its body to follow a winning path only when it has recent information in its associated ping direction.

In the situation where none of the action selection neurons fire a spike in a certain time window after the outgoing ping, all paths are undesirable for the bat to follow and the emergency "turnaround" maneuver will be executed as described in Section Motor Control. It is achieved by adding an "emergency" neuron in the action selection layers with a fixed spike latency and connecting it to a motor neuron responsible for the emergency maneuver. The emergency neuron will be inhibited if any other action selection neurons fire a spike before it does, creating a time window where a path is good enough to follow.

EXPERIMENT RESULTS

Simulation of the Analog Model

We focus our simulation on the bat traveling in a field with densely placed obstacles to test the capabilities and characteristics of the analog model. 1,400 identical obstacles are placed randomly in a 50×50 m field with two-dimensional periodic boundary conditions (**Figure 10A**). The objective of the bat is to travel in the leftward direction without colliding with any obstacles. We provided a dynamic goal input to drive the bat to a goal destination at the left boundary ($X = 0$) with the same vertical position (same Y coordinate) and updated it at each time step according to the position of the bat. When the bat crosses the left boundary, it will reappear at the right boundary with the same orientation and speed before the crossing. An example of the traveled path in a simulation is shown in **Figure 10B** and it shows that the bat visited most of the viable paths instead of repeatedly traveling on a few paths.

In the simulations, the bat used the accurate locations of the obstacles whose centers were in the current FOV. The occlusion of distant obstacles was not simulated. The FOV function (detectable distance vs. relative angle to the head direction) used in the simulation is a Gaussian distribution with $\sigma = 30$ with a maximum distance $R_{\text{MAX}} = 5\ m$, resulting in a half-maximum width of 70.7. For big brown bats, the half-power beamwidth (the angular width of the beam pattern at the 3 dB cutoff points) of their emitted ultrasonic signal at 35–40 kHz is 56 to 80 (Ghose and Moss, 2003; Gaudette et al., 2014). Although the FOV of a typical sonar system is not equivalent to the beampattern of the emitted sound, we approximated them to be similar and designed the FOV with a similar shape. The bat emitted sonar pings at a maximum rate of 5 Hz in the simulation with the exception in section 4.1.4 where the maximum sensing rate is increased to test its influence on the performance. Big brown bats can send out sonar pulses to sense obstacles at a rate of up to 90 Hz (Sändig et al., 2014) and turn their heads to ping in a different direction in <30 ms (Surlykke et al., 2009). We used a much lower maximum sensing rate in the simulation to push the limits of the model and test its capabilities. It is assumed that there are no errors in the steering action and the bat was able to follow the winning path exactly.

Defining performance metrics for collision avoidance is dubious because it must always be evaluated in the context of

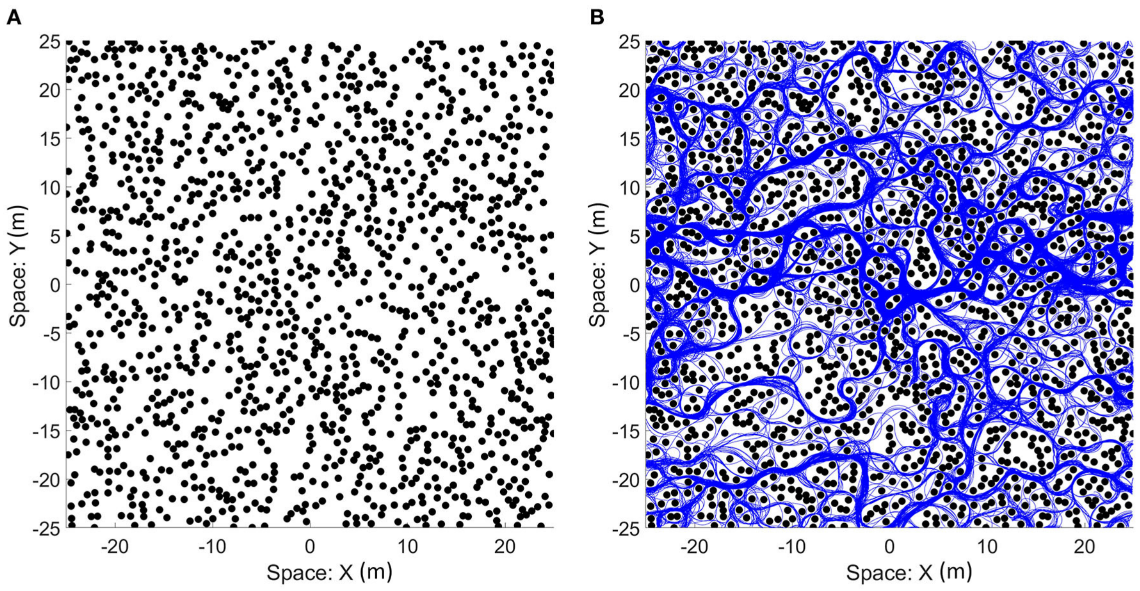


FIGURE 10 | (A) An example of the field with randomized locations of obstacles. One thousand four hundred obstacles (black circles) were placed in a 50 × 50 m area. The size of the obstacles shown in the figure is the size of the zone of collision (combination of the sizes of the bat and the obstacle). The space of the field can be mapped onto a torus to simulate a bat traveling on an infinite plane as the bat can sense and travel through boundaries. **(B)** The paths that the bat traveled (blue line) on the field. Although open areas were traveled more often by the bat, most of the viable paths were visited.

another movement-inducing behavior (e.g., goal-seeking) and most comparisons are done between performance measures such as path length and travel time. Furthermore, collision avoidance performance is strongly correlated with the limitations of the sensors. Comparisons to other algorithms are problematic if their algorithms do not also use narrow-FOV sensors. Nonetheless, we have defined a performance measure to understand the impact of features and parameters in the proposed model. We use the goal-seeking behavior and measure how long the bat can move toward the goal without a collision. Specifically, the number of unique obstacles that the bat detected, but did not collide with, is counted as the number of avoided obstacles. Only unique obstacles were counted because the bat could be trapped temporarily in a local area and the obstacles around the bat should not be counted more than once in this scenario. Whenever the bat crossed the left boundary, the bat was considered to have entered a new environment and all the obstacles could be counted again. The simulation ends if one of the three scenarios happened: (1) the bat collided with an obstacle; (2) the bat has not crossed the left boundary in a long time (2,500 s) suggesting that it is trapped; (3) the simulation has reached a time limit (250,000 s). In the second and third scenarios, the simulation restarts with a different random seed, and the number of avoided obstacles is accumulated. Whenever the bat collided with an obstacle, the accumulated number of avoided obstacles before the collision is recorded and is considered as a sample.

Density of Obstacles

To test how the proposed model performed with different densities of obstacles in the field, simulations were run with different numbers of obstacles ranging from 500 to 1,700. Because the simulated area was constant, changing the number of obstacles in the 50 × 50 m field is equivalent to changing the density of obstacles. The simulation results are shown in **Figure 11A**. As expected, the performance increased as the density decreased.

Desirability Function

To test the effect of different terms in the desirability function (Equation 6) under different maximum speed settings, we compared the performance between a control configuration of the proposed model across different speed settings (i.e., control group), a configuration without WTA hysteresis and a configuration without ping direction bias (**Figure 11B**). The control simulations used all the terms of the desirability function described in Equation 6, while the group without WTA hysteresis excluded the hysteresis term ($H = 0$) and the last group excluded the ping direction bias term ($P = 0$). The average number of avoided obstacles is calculated based on 200 samples under different maximum speed (V_{MAX} in Equation 1) settings and the 95% confidence intervals for each configuration are shown as error bars in **Figure 11B**.

The group without WTA hysteresis had significantly worse performance ($p < 0.05$) only at $V_{\mathrm{MAX}} = 2\ m/s$ compared to the control group. Hysteresis is generally introduced in a WTA layer to prevent the winner from oscillating between a few similar candidates due to noise in the system. We believe that one of the main reasons why WTA hysteresis did not have a significant impact on the performance is because sensory or other noise is not included in the simulation.

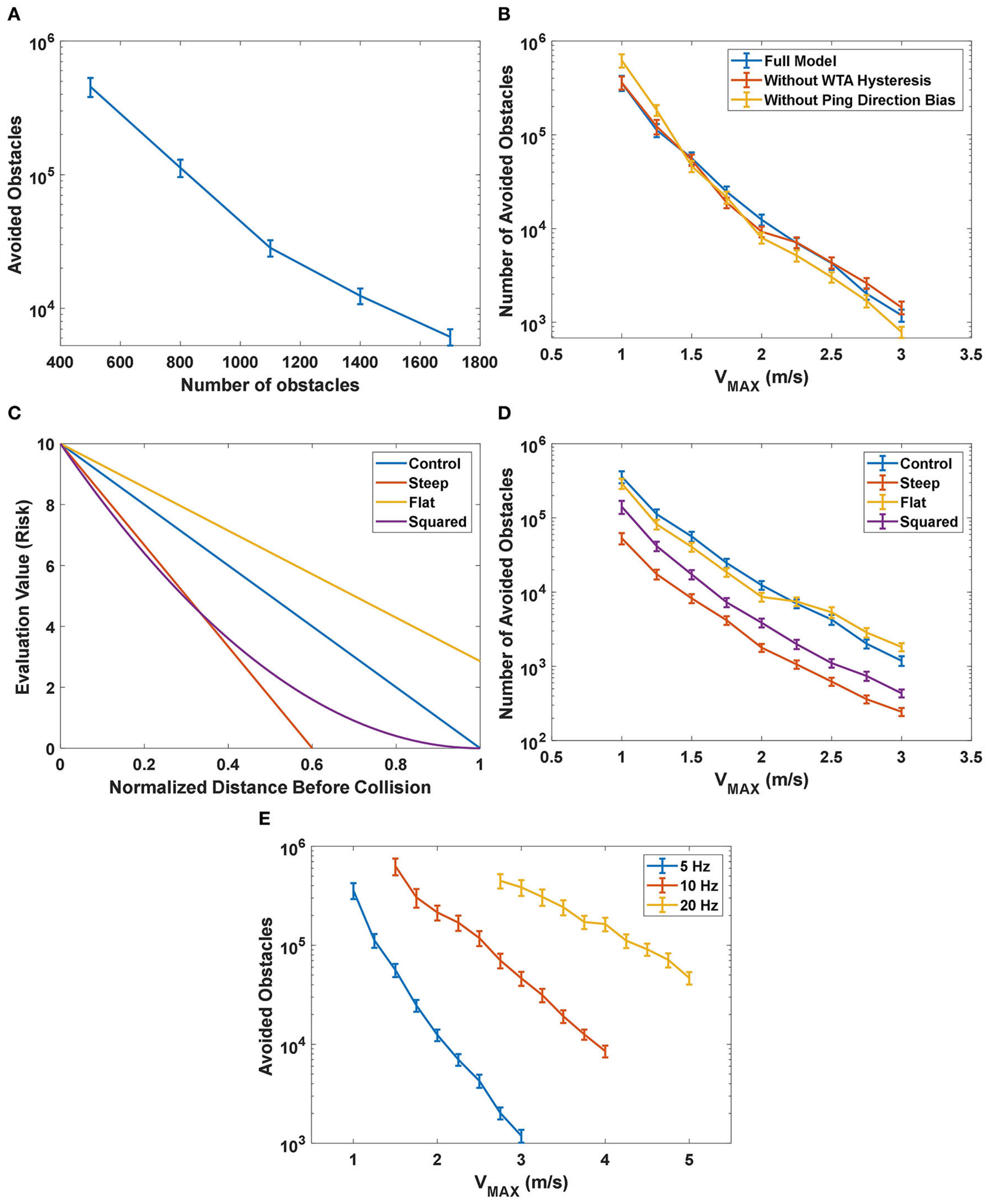


FIGURE 11 | (A) The performance of the proposed model as a function of the number of obstacles in the field. The number of avoided obstacles increased, approaching infinity, as the number of obstacles decreased. **(B)** The simulation results of the effect of different terms in the desirability function under different maximum speed (V_{MAX}) settings. The control configuration included all the terms described in Equation 6 while the configuration without WTA hysteresis had $H = 0$ and the group without ping direction bias had $P = 0$. The group without WTA hysteresis had similar performance at most of the speed settings compared to the control group, while the group without ping direction bias had significantly better performance at lower speeds ($V_{MAX} \leq 1.25\ m/s$) and significantly worse performance at higher speeds ($V_{MAX} \geq 2\ m/s$). **(C)** The shape of different immediacy functions tested in the experiment. **(D)** The performance of the model with different immediacy functions. The results from the "Steep" group and the "Squared" group showed that reducing the significance of distant obstacles negatively impacted the performance across all speed settings, and thus indicated that distant obstacles were important to the decision-making process of the model. The comparison between the control group and the "Flat" group showed that increasing the significance of distant obstacles could be beneficial at higher speed. **(E)** The effect on the performance from changing the maximum rate of sonar pings. The performance not only was significantly increased with increased rate of pings but also declined more slowly with the increase in the maximum speed.

A comparison of performance shows that the group without ping direction bias had significantly better performance ($p < 0.05$) than the control group when $V_{\mathrm{MAX}} \leq 1.25\ m/s$ and has significantly worse performance ($p < 0.05$) when $V_{\mathrm{MAX}} \geq 2\ m/s$. As mentioned in section action selection layer, the function of the ping direction bias is to encourage the bat to select the path with the most recent information when several paths have similar risks. At low speeds, the evaluation memory of the paths from different ping directions is relatively accurate and adding the ping direction bias frequently prevents the bat from selecting the actual "best" path, thus dropping the performance. At high speeds, however, the evaluation memory may be wildly inaccurate and selecting a slightly better path according to the memory could be more dangerous than selecting a "good" path with recent and accurate information. The results suggest that features in the desirability function provide benefits in different situations and adding a speed-dependent controller on the weights of different features could help improve performance.

Immediacy Function

We tested and compared different functions representing the immediacy of avoiding an obstacle. As a reminder, the immediacy of avoiding a blocking obstacle j on a path p is represented as a function (f).

$$f_p(j) = E_{\mathrm{MAX}} \cdot \left[1 - \frac{r_p{}'(j)}{R_{\mathrm{MAX0}} \cdot \gamma_p}\right] \tag{11}$$

where E_{MAX} is a constant representing the maximum value of f_p and is set to 10 for all the different configurations in this experiment. $r_p{}'(j)$ is the distance that the bat can travel along path p before colliding with a blocking obstacle j. The distance is then normalized with the speed profile γ_p to represent the time before colliding with the obstacle.

The results from this experiment are shown in **Figures 11C,D**. The control group had the same function of Equation 11 with $R_{\mathrm{MAX0}} = R_{\mathrm{MAX}} = 5$ and it decreased linearly with the distance to the obstacle normalized by the speed profile γ_p. Because $R_{\mathrm{MAX0}} = R_{\mathrm{MAX}}$, the immediacy function decreased to 0 when the normalized distance $\frac{r_p{}'(j)}{\gamma_p}$ is 1, as described earlier. The "Steep" group had $R_{MAX0} = 3$, thus creating a steeper linearly-decreasing function compared to the control group. The "Steep" function was rectified to 0 when it dropped below 0, which means that more distant obstacles were simply ignored. The "Flat" group had $R_{MAX0} = 7$, resulting in a flatter function that included the contributions of a blocking obstacle even when it was distant. As a tradeoff, the differences between obstacles at different distances were reduced. The "Squared" group had a function as $f_p(j) = E_{\mathrm{MAX}} \cdot \left[1 - \frac{r_p{}'(j)}{(R_{\mathrm{MAX}} \cdot \gamma_p)}\right]^2$, which emphasizes close obstacles without completely ignoring distant obstacles.

As shown in **Figure 11D**, the number of avoided obstacles in the "Steep" group was significantly lower than all other groups across all speed settings, followed by the "Squared" group with the second-worst performance. It showed that throwing away or reducing the significance of the information about distant obstacles severely reduced performance, indicating that the proposed model could make good use of this information in its decision-making process. The "Flat" performed significantly worse ($p < 0.05$) when $1.5\ m/s \leq V_{\mathrm{MAX}} \leq 2\ m/s$ but performed significantly better ($p < 0.05$) at speeds higher than 2 m/s. Distant obstacles present a greater threat at higher speeds, thus the steering decisions could benefit from emphasizing them. The results suggest that adjusting the evaluation function accordingly at different speed settings could improve performance.

Maximum Ping Rate

In the previous experiments, the maximum ping rate was set to be 5 Hz. As mentioned before, echolocating bats can emit outgoing sonar pings up to a rate of 90 Hz (Sändig et al., 2014). In this experiment, we increased the maximum rate of sonar pings to 10 and 20 Hz to test its effect on the performance.

As shown in **Figure 11E**, the performance improved significantly with increased ping rates. Increasing the ping rate also helped their performance decline more slowly with the increase in speed settings, indicated by a lower slope of 10 and 20 Hz curves compared to that of the 5 Hz curve. A higher rate of pings allowed the bat to travel at a much higher speed while maintaining its performance.

Spiking Neural Model Simulation

The proposed spiking neural model was implemented in Python using a combination of Hodgkin–Huxley neurons (Hodgkin and Huxley, 1952) and leaky integrate-and-fire (LIF) neurons. The Hodgkin-Huxley model was applied to as many neurons as possible to show the biological plausibility of the proposed spiking neural network, and simpler LIF neuron models were used to implement sensory neurons in the sensory layer, integrating neurons in the evaluation memory and WTA neurons in the action selection layer. The sensory neurons were implemented in LIF neurons to speed up the simulation because of their large number and their simple function (fire a spike upon receiving a spike). Implementing them with the LIF neurons does not change their function significantly compared to the implementation with Hodgkin-Huxley neurons. The integrating neurons and the WTA neurons used LIF models because their function includes integrating input spikes over a longer time interval (~40 ms). In total, there are 2,931 LIF neurons and 2,078 Hodgkin-Huxley neurons in the neural model.

The synaptic connections to the Hodgkin-Huxley neurons were implemented with a time-dependent conductivity function $g_{\mathrm{syn}}(t)$. The synaptic current is calculated as

$$I_{\mathrm{syn}}(t) = g_{\mathrm{syn}}(t) \cdot [v(t) - E_{\mathrm{syn}}] \tag{12}$$

The reversal potential E_{syn} and the function $g_{\mathrm{syn}}(t)$ can be used to describe different types of synapses. For excitatory synapses, E_{syn} is set to 10 mV whereas for inhibitory synapses, E_{syn} is set to −70 mV. In this simulation, an alpha function was used to describe the synaptic conductance $g_{\mathrm{syn}}(t)$ as:

$$g_{\mathrm{syn}}(t) = \bar{g}_{\mathrm{syn}} \cdot (t - t_0)(\frac{e}{\tau})e^{-\frac{(t-t_0)}{\tau}} \cdot \Theta(t - t_0) \tag{13}$$

where τ is the time constant, $\bar{g}_{\mathrm{syn}}$ is the maximum conductance, t_0 is the arrival time of a pre-synaptic spike and $\Theta(t)$ is the Heaviside step function. The synaptic connections to LIF neurons are simplified as current injections, $I_{\text{syn\_LIF}}(t)$, which are described by a boxcar function as

$$I_{\text{syn\_LIF}}(t) = \bar{I}_{\text{syn}} \cdot [\Theta(t - t_0) - \Theta(t - t_0 - T_{\text{syn}})] \tag{14}$$

where $\bar{I}_{\text{syn}}$ is the amplitude of the synaptic current and T_{syn} is the duration of the synaptic input.

Each sensory map in the sensory layer consists of 471 neurons representing locations of up to 25 ranges at 0.2-meter increments (from 0.2 to 5 m) and 31 angles at 4-degree increments (from -60° to 60°). Big brown bats (*Eptesicus fuscus*) are shown to be able to discriminate range differences as small as 1 cm (Simmons, 1973) and angular differences of 6° (Peff and Simmons, 1972), while other species such as the greater spear-nosed bat (*Phyllostomus hastatus*) can differentiate objects that are 4° apart. We used a relatively small number of neurons to represent the range because a higher range resolution significantly increased the total number of neurons and slowed the simulation while not significantly improving the performance of obstacle avoidance. The neurons that would represent locations outside of the field-of-view of the sonar were removed, thus the number of neurons in each map is smaller than the total number of possible combinations of ranges and angles. During the simulation, if an obstacle is within the sonar field-of-view when the bat pings, the obstacle will cause the sensory neuron with the closest receptive field in the head map to fire a spike.

The simulation of the proposed spiking neural network was implemented in Python and run on a CPU (Intel Xeon Cascade Lake) with a time step of 0.02 ms. A small time step was needed to ensure the correct simulation of the Hodgkin-Huxley neurons. The synaptic weights were fixed and calculated before the start of the simulation. An example of the simulation is shown in **Figure 12**. After the first sonar ping, the EM combined information from the sensory layer and produced three groups of spikes (blue dots in **Figure 12B**) whose latencies represent the risk values of the paths that were blocked by the three detected obstacles. Smaller latencies from the EM represent larger risk values. The EM spikes with smaller latencies caused larger delays in the firing of the WTA neurons (gray vertical lines), making those paths less desirable and less likely to win in the temporal WTA. Because several neurons in WTA layer 1 fired simultaneously, there were multiple winners (red dots). The biased WTA (layer 2) selected the winners that were closer to the center and in the same ping direction group (black dots).

The spikes from the winners, which belong to the middle-left group, caused the middle-left head motor neuron to fire a spike (not shown). As a result, the bat turned its head to the middle-left direction for the next sonar ping (right panel in **Figure 12A**). Because the bat did not have recent sensory information in the middle-left direction, no body motor neurons in the BML fired and the bat kept its original trajectory. The bat also maintained the memory of the spike latencies from the detected obstacles in the EM.

After the second sonar ping at 200 ms, the output neurons in the EM fired spikes in response to the two newly detected obstacles (EM neurons 4–5 and 11–13). At the same time, the neurons reproduced the spike latencies from the previously detected obstacles that were no longer in the field of view (EM neurons 16–17 and 22–26). Because the three winners from the action selection layer were all from the middle-left direction and the bat had recent information in this direction, the BML neurons fired spikes (green dots) and the bat executed the path averaged from the fired BML neurons (right panel in **Figure 12A**).

The membrane potentials of the neurons in an activated latency memory unit in this example are shown in **Figure 12C** to demonstrate the mechanism of the evaluation memory. The A neuron was excited by a delay line neuron (not shown) and fired a spike with a fixed latency after every sonar ping. Although the B neuron was excited by the same delay line neuron, it did not fire after the first sonar ping because it was inhibited by the tonically firing inhibitory interneuron (INH). Because a blocking obstacle was detected after the first sonar ping, the integrating neuron (INT) received a spike from the sensory layer and produced a spike with a latency representing the immediacy of avoiding that obstacle. When the spike from the INT neuron and the spike from the A neuron arrived at the coincidence detector (CD) neuron within 3 ms of each other, the CD neuron fired a spike which excited the output neuron of the evaluation memory unit. The spike from the CD neuron also inhibited the INH neuron and suppressed its firing for a long duration ($\sim$150 ms), releasing neuron B from its inhibition.

After the second sonar ping, the B neuron fired a spike at the same time as the A neuron because they were excited by the same delay line neuron. The coincidence of the spikes from A and B neurons caused the CD neuron to fire again and thus the same spike latency was reproduced. The spike from the CD neuron suppressed the INH neuron again and "extended" the latency memory. In this example, because the bat made a motor decision after the second sonar ping, all the latency memories were reset by an excitatory spike to the INH neuron ("reset spike" in the bottom panel in **Figure 12C**) that was strong enough to overcome the previous inhibition from the CD neuron and caused the INH neuron to start firing again. Once the INH neuron is firing, the B neuron is suppressed, and the CD neuron will no longer fire without the input from the INT neuron. Hence, the latency memory is "erased."

The neural network was also simulated in the same environments used in the simulation of the analog model described in Section Simulation of the Analog Model. One of the simulation results is shown in **Figure 13** where the bat successfully reached the left boundary 4 times without colliding with any obstacles.

Due to the complexity of the Hodgkin-Huxley model, simulating the proposed spiking neural network on a CPU is extremely slow ($\sim 10^5$ times slower than simulating the analog version), and collecting enough data to do performance analysis as in Section Simulation of the Analog Model is impractical with the current implementation. The short simulation run shown in **Figure 13** took more than 5 days to complete. A neuromorphic VLSI implementation, however, could vastly improve the running speed of the proposed spiking neural network and can allow it to run in real-time on a mobile

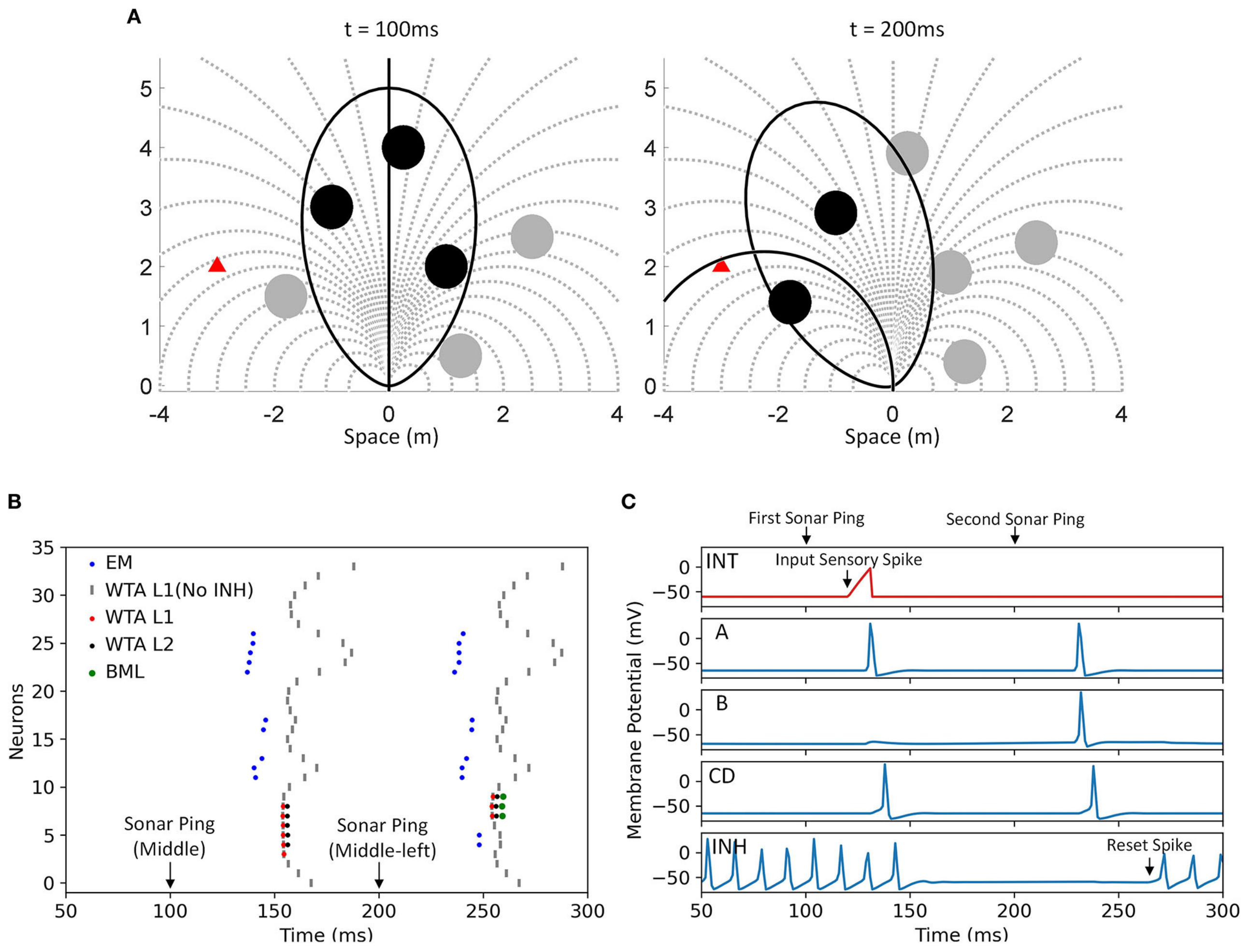


FIGURE 12 | (A) The bat is in an environment with 6 obstacles (filled circle) and a desired destination (filled triangle in red) without prior information of the environment. The environment is the same as the example given in **Figure 5** for a clearer comparison between the analog model and the spiking neural model. It is assumed that it was flying along a straight path (solid straight line) and it directed its first sonar ping to the front (ellipsoidal FOV shown in solid line). The circles of the obstacles are drawn with the size of the zone of collision to show the paths that they are blocking. The bat has a limited selection of motor choices (dotted line, 33 in total) and each of the motor actions corresponds to a neuron in the spike raster plot below. The bat sent the first sonar ping at $t = 100$ ms and a second one at $t = 200$ ms. The detected obstacles (filled circle in black) from each of the sonar ping caused the corresponding sensory neurons to fire. **(B)** The spikes from the output neurons in the evaluation memory (EM), the WTA neurons in the action selection layer and the neurons in the body motor layer (BML) were shown. The spike latency of each EM neuron represents the "risk" value of a path, and the spike latency of each WTA neuron represents the desirability of a path. The neurons with smaller indices represent paths to the left and the neurons with larger indices represent paths to the right. The spikes from an imaginary WTA layer 1 without recurrent inhibition are also plotted (gray vertical line) for a better demonstration of the impact that the spikes from the EM had on the action selection layer. **(C)** The membrane potentials of the neurons in the 9th latency memory unit (LMU) and the integrating neuron (INT) for path 24 are shown to demonstrate the working mechanism of the evaluation memory. The integrating neuron was implemented as a LIF neuron (red curve) and the neurons in the LMU (neurons A, B, CD and INH) were implemented using Hodgkin-Huxley model (blue curve).

platform. It is important to note that it is not necessary to use the Hodgkin-Huxley model in the neural network and it is only used to show the biological plausibility of the network with more realistic neuron characteristics. The proposed network can produce similar results using simpler neural models.

Robot Implementation

We have implemented the analog version of the Curved Openspace model on a mobile robot with a car-like steering mechanism (**Figure 14**). The 3-D printed sonar head (white) is mounted on a servo motor at the front of the vehicle and has three vertically stacked sonar transducers, each facing a different direction. To collect sensory data, the middle transducer sends out a sonar ping and the echoes reflected from objects are measured by all three transducers. The distances to objects are measured using the time of flight of their echoes, and the direction of each object is computed using the relative amplitudes of the sonar echo. The custom sonar transducer boards output a logarithmically-compressed envelope

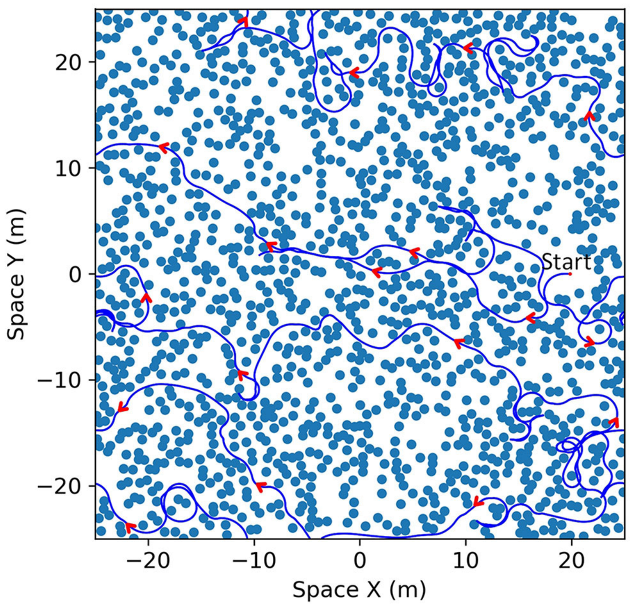


FIGURE 13 | An example of the bat traveling at a maximum speed of 2 m/s in a dense forest using the proposed spiking neural model with a spike-timing representation. One thousand four hundred obstacles (filled circles in blue) were placed in a 50 × 50 m area. The size of the obstacles shown in the figure is the size of the zone of collision (combination of the sizes of the bat and the obstacle). The parameters of the bat were kept the same as the parameters used in the simulation of the analog model in section simulation of the analog model, except for the maximum ping rate which was set to 10 Hz in this simulation. The field has two-dimensional periodic boundary conditions to allow the bat to travel as if in an infinitely large space. A goal input is provided to drive the bat to a goal destination at the left boundary (X = 0) with the same vertical position (same Y coordinate). The paths that the bat traveled on the field are shown as blue curves.

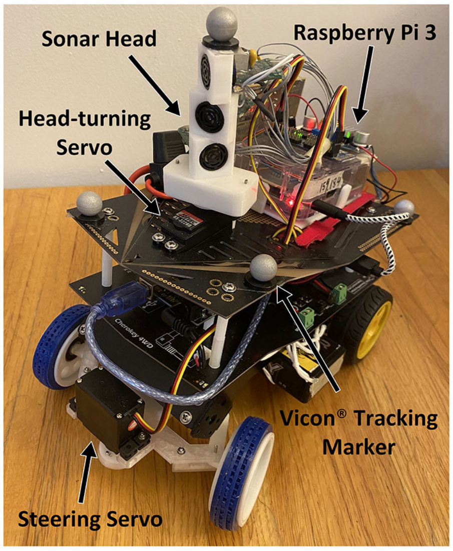


FIGURE 14 | The mobile robot on which the analog model is implemented. The robot consists of a sonar head with three sonar transducers mounted on a servo motor, a Raspberry Pi 3 running the analog model, an Arduino Uno board controlling the rear wheel DC motors, a servo motor controlling the front wheels with an Ackermann steering geometry, an Adafruit 16-channel servo driver, and a lithium polymer (LiPo) battery. The sonar transducers are custom-modified MaxBotix sonar transducers (piezo) that act as both a speaker and a microphone. They resonate at 40 kHz and will only detect signals near this frequency. The transmitted beam has a half-power beamwidth of about 60°. The top transducer points 45° to the right (from the head's point of view), the middle transducer points straight forward, and the bottom transducer points 45° to the left. The trained radial basis function (RBF) network and the analog version of the Curved Openspace model are implemented in Python on a Raspberry Pi 3 mounted on the back of the robot. The Raspberry Pi allows for wireless operation via WiFi and coordinates the commands to a servo controller board (for sonar head pointing and steering) and the powered rear drive wheels. The servo motor in the front of the vehicle controls the turning rate of the vehicle using an Ackermann steering mechanism. A given steering angle thus translates to a specific turning radius of the vehicle and a lookup table is used during operation. Because of the limited resolution of the steering motor, the total number of motor actions was reduced from 33 to 15 in the robot implementation. The neural model controls the speed of the vehicle by adjusting the power delivered to the rear wheel DC motors using pulse-width modulation.

signal of echoes as an analog voltage. For each echo, the peak voltage on each of the three transducers is recorded by a microcontroller-based analog-to-digital converter (ADC). A radial basis function (RBF) network is then used to map the echoes to angles in the horizontal plane at 1-degree increments. The RBF network was trained using a dataset consisting of 910 echoes from 91 different angles and 10 intensities (to simulate different ranges).

The program sends out a sonar ping from the middle transducer, waits for 3 ms for the outgoing pulse to die out before collecting sonar echoes and then uses the RBF network to localize the detected obstacles. The goal input is provided in the straight-forward direction initially and it can change according to the movement of the robot to guide it toward the initial goal direction. The risk values and the desirability of 15 different paths are then calculated and a winning path along with the next ping direction is executed. Finally, the program waits for the turning of the head to finish before sending out a new sonar ping and starts the cycle again. If no head movement is required, a sonar ping is sent in the current ping direction immediately. Because of the limited rotational speed of the head-turning servo, the ping frequency varies from 3 Hz when the head needs to turn from one end to another, to 12 Hz when no head movement is required.

The mobile robot was tested in different forests of obstacles and its positions were recorded in 3-dimensions using a Vicon® marker-based visual tracking system. The robot path and ping directions of the robot in three different example forests are shown in **Figure 15**. In the first configuration (**Figure 15A**), when the robot did not detect any obstacles near the middle path, it did not need to turn the head to other directions because the goal direction was in the middle and the middle path was the winner. As a result, most of the time the robot kept pinging down the center. The sonar pings to the side were caused by errors in the sonar RBF network that sometimes incorrectly localized side obstacles to the center. In the second configuration (**Figure 15B**),

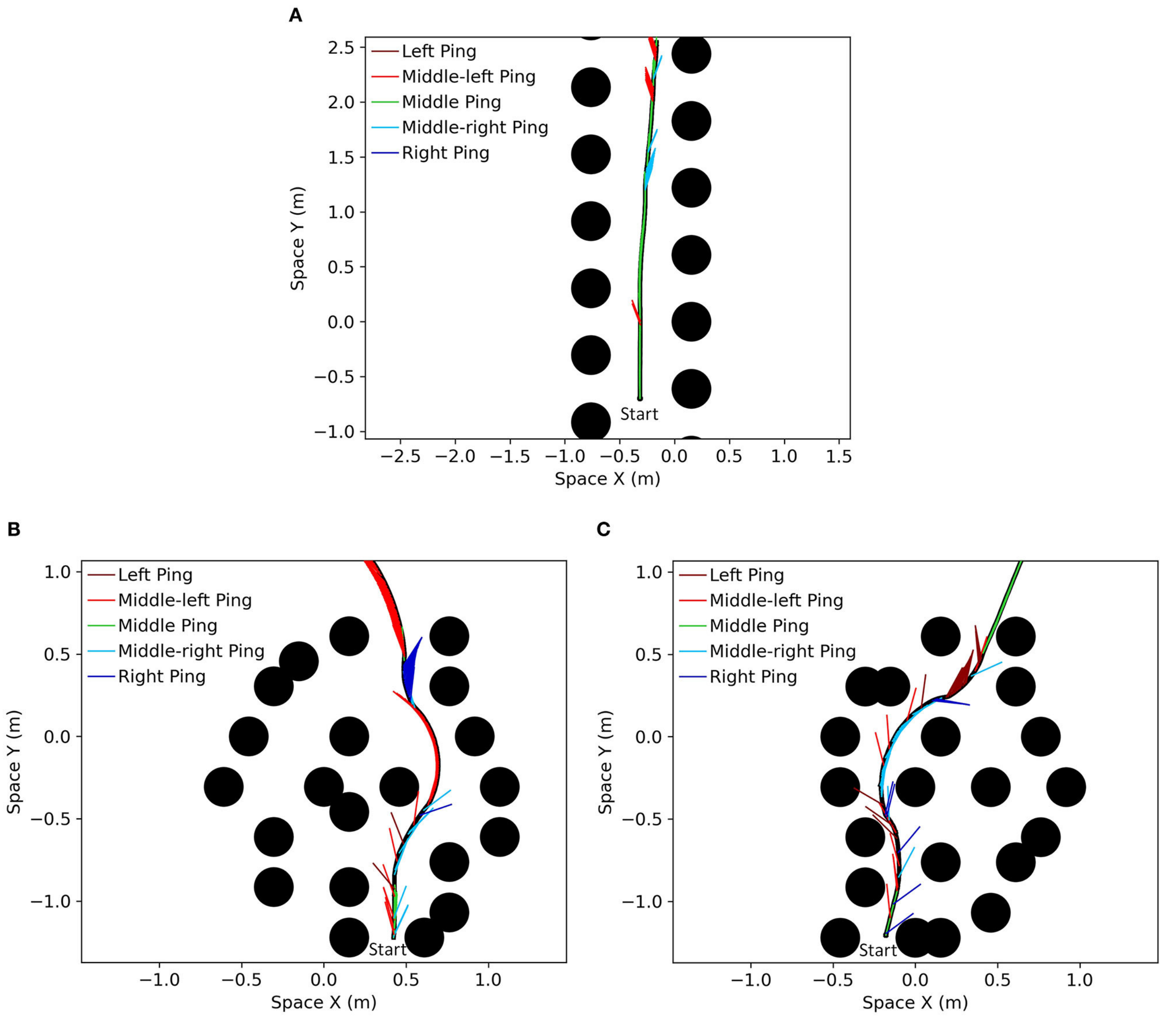


FIGURE 15 | The path and ping directions of the robot during runs in three different forests. The obstacles (5 cm diameter PVC poles) are shown as black circles and the size of the circle is the size of the zone of collision (size of the robot plus the size of the PVC poles). The path the robot took is shown as a solid black line and the ping directions are shown as solid-colored lines pointing at the corresponding ping directions. **(A)** Robot running down a corridor formed by two lines of evenly spaced obstacles. The ping directions in this run were mostly down the center (green solid lines). **(B)** The robot traveling in a forest composed of 21 obstacles placed in a 1.8 × 1.8 m area. The robot explored other paths more often by pinging in different directions in more cluttered areas, whereas it mainly pinged in the direction of its current path in the more open areas. **(C)** The path of the robot in a denser forest consists of 21 obstacles in a 1.4 × 1.8 m area.

the area around the starting point of the robot is more cluttered, so the robot pinged around and explored different paths more often. In the second half of the run, the robot entered a more open area where it mostly pinged in the direction that it was traveling in. A similar result can be observed in a denser forest example (**Figure 15C**). The video recordings of the two runs shown in **Figures 15B,C** are provided as **Supplementary Materials**.

Several problems can occur in a real sonar system compared to the simulation. First, occlusion can happen when one object is directly behind another object and the sonar system cannot detect the far object accurately or at all. Another problem that can occur are multi-path echoes where the sound is bouncing between different objects, creating additional echoes with different delays that might be erroneously interpreted as different objects. During the testing of the robot, the problems mentioned above did not create significant difficulties, given our simplified environment, but we acknowledge that this might not be the case in complex 3D environments. One of the problems that we encountered, however, was the near-field *blind zone*. Following an outgoing ping, the highly resonant transducers will produce a decaying signal that continues for some time, interfering with amplitude measurements of echoes from nearby objects. Although objects can be *detected* within this signal, amplitude measurements for localization are not practical. Its range can be reduced by lowering the intensity of the outgoing ping when the robot is encountering close obstacles. Another problem that we encountered was that in certain cases, the positions of the obstacles calculated by the RBF network were inaccurate and caused the robot to steer in an undesirable direction. The incorrect motor actions were often quickly corrected in the following sensor cycle once the positions of the obstacles were accurately calculated and the incorrect memory was replaced by the accurate sensory information.

DISCUSSION

This paper presents an obstacle avoidance neural model that provides steering decisions to a sonar-guided "bat" to avoid static obstacles while pursuing a goal. To achieve better matching between the desired path and a path that the bat can realistically follow, the Curved Openspace model projects the sensory-based obstacle data into "motor space" before comparing motor choices. Although the idea of selecting motor actions instead of sensory directions has been proposed before (Simmons, 1996; Fox et al., 1997), they used sensors with a wide field-of-view and did not need to consider the question of how to gather information efficiently over time. Taking into account the limited field-of-view of an echolocating bat or a practical sonar system, an attentional system is proposed to control the direction of sonar pings in a time and energy efficient way where the bat will ping in the most beneficial direction to guide its motor action selection. The fact that the bat is moving while gathering and integrating information introduces problems such as the inaccuracy of old data. To alleviate the problem of inaccurate memory, we designed an evaluation memory that decays old data and a desirability function that incorporates both the information about the goal and obstacles as well as the recency of the sensory data. The presented simulations showed the effectiveness of different configurations under different situations. Although the proposed neural model is described as a model for sonar-guided creatures or vehicles, it can be driven with other sensors such as cameras or laser rangefinders.

The analog model was implemented in real-time on a car-like robot with an active sonar system. The robot was tested in multiple scenarios and the results showed that the analog model works with real sonar sensory data. During these runs, the robot would only turn its head and collect sensory data in non-traveling directions when necessary, showing the efficiency of the proposed attentional system. It is important to note that the point of the robot implementation is to demonstrate that the proposed neural model operates as expected with a real sonar and an inexpensive robot in closed-loop behavior.

A spiking neural network using spike-timing representations is also described in the paper. Instead of using a large population of neurons to represent analog signals with spike rates, we used spike-latency to represent signal values for computation. Sonar systems inherently utilize a timing code where low echo latency represents a close object and high echo latency represents a distant one. Only a small adjustment on this latency is required to translate this into the immediacy of avoidance measure. Moreover, a sonar system will receive echoes from close obstacles earlier than distant obstacles. This means that a creature implemented with the proposed spiking neural model could reduce the time to make a motor decision if it chooses to focus only on closer obstacles in a cluttered environment. With the "race-to-first-spike" WTA mechanism, an early decision can be forced by increasing the passive excitatory input to further reduce the time spent on decision-making if most of the computation happens early. The presented simulations showed that the bat running on the proposed neural network can navigate through dense forests. The slow simulation speed of the neural network, however, prevents the current simulation from running extensive performance analyses and advocates for implementation on neuromorphic VLSI hardware. Overall, the use of input spike timing to modulate the efficacy of a synaptic connection can be an effective mechanism that does not rely on increasing spike rates or modulating synaptic strength. Although the Curved Openspace model and its spiking neural model are proposed for obstacle avoidance on a 2-D plane in this paper, it would not be difficult for the neural model to operate in the 3-D world by extending the motor actions to 3-D trajectories.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

CW contributed to designing and simulating the algorithm, testing the robot, designing and simulating the spiking neural network, and manuscript preparation. TH contributed to designing the algorithm, designing the sonar system, designing the robot, designing the spiking neural network, and manuscript preparation. All authors contributed to the article and approved the submitted version.

FUNDING

This work was supported by the U.S. Office of Naval Research (N000141210339), the U.S. Air Force Office of Scientific Research (FA9550-14-1-0398, Center of Excellence), the U.S. National Science Foundation (SMA1540916, Neuromorphic Engineering Workshop), and the Maryland Industrial Partnerships Program (MIPS 2019-2021).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot.2022.850013/full#supplementary-material

REFERENCES

Aggarwal, S., and Kumar, N. (2020). Path planning techniques for unmanned aerial vehicles: a review, solutions, and challenges. *Comput. Commun.* 149, 270–299. doi: 10.1016/j.comcom.2019.10.014

Borenstein, J., and Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Trans. Syst. Man Cybern.* 19, 1179–1187. doi: 10.1109/21.44033

Borenstein, J., and Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Trans. Robot. Autom.* 7, 278–288. doi: 10.1109/70.88137

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. MIT press. p. 1–94.

Bry, A., Bachrach, A., and Roy, N. (2012). "State estimation for aggressive flight in GPS-denied environments using onboard sensing," in *2012 IEEE International Conference on Robotics and Automation* (St. Paul, MN: IEEE), 1–8. doi: 10.1109/ICRA.2012.6225295

Bry, A., and Roy, N. (2011). "Rapidly-exploring random belief trees for motion planning under uncertainty," in *2011 IEEE International Conference on Robotics and Automation* (Shangai: IEEE), 723–730. doi: 10.1109/ICRA.2011.5980508

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* 4, 23–33. doi: 10.1109/100.580977

Gaudette, J. E., Kloepper, L. N., Warnecke, M., and Simmons, J. A. (2014). High resolution acoustic measurement system and beam pattern reconstruction method for bat echolocation emissions. *J. Acoust. Soc. Am.* 135, 513–520. doi: 10.1121/1.4829661

Ghose, K., and Moss, C. F. (2003). The sonar beam pattern of a flying bat as it tracks tethered insects. *J. Acoust. Soc. Am.* 114, 1120–1131. doi: 10.1121/1.1589754

Häusser, M., Raman, I. M., Otis, T., Smith, S. L., Nelson, A., Du Lac, S., et al. (2004). The beat goes on: spontaneous firing in mammalian neuronal microcircuits. *J. Neurosci.* 24, 9215–9219. doi: 10.1523/JNEUROSCI.3375-04.2004

Hodgkin, A. L., and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* 117, 500–544. doi: 10.1113/jphysiol.1952.sp004764

Horiuchi, T. K. (2009). A spike-latency model for sonar-based navigation in obstacle fields. *IEEE Trans. Circuits Syst. I Regular Pap.* 56, 2393–2401. doi: 10.1109/TCSI.2009.2015597

Itti, L., and Koch, C. (2000). A saliency-based search mechanism for overt and covert shifts of visual attention. *Vis. Res.* 40, 1489–1506. doi: 10.1016/S0042-6989(99)00163-7

Khatib, O. (1986). "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles* (New York, NY: Springer), 396–404. doi: 10.1007/978-1-4613-8997-2_29

Latombe, J. C. (2012). *Robot Motion Planning*, *Vol. 124*. Springer Science and Business Media.

Lee, C., Rohrer, W. H., and Sparks, D. L. (1988). Population coding of saccadic eye movements by neurons in the superior colliculus. *Nature* 332, 357–360. doi: 10.1038/332357a0

Lyu, H., and Yin, Y. (2019). COLREGS-constrained real-time path planning for autonomous ships using modified artificial potential fields. *J. Navigation* 72, 588–608. doi: 10.1017/S0373463318000796

Milde, M. B., Blum, H., Dietmüller, A., Sumislawska, D., Conradt, J., Indiveri, G., et al. (2017). Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system. *Front. Neurorobotics* 11:28. doi: 10.3389/fnbot.2017.00028

Molinos, E., Llamazares, Á., Ocaña, M., and Herranz, F. (2014). "Dynamic obstacle avoidance based on curvature arcs," in *2014 IEEE/SICE International Symposium on System Integration* (IEEE), 186–191. doi: 10.1109/SII.2014.7028035

Molinos, E. J., Llamazares, A., and Ocaña, M. (2019). Dynamic window based approaches for avoiding obstacles in moving. *Robot. Auton. Syst.* 118, 112–130. doi: 10.1016/j.robot.2019.05.003

Peff, T. C., and Simmons, J. A. (1972). Horizontal-Angle resolution by echolocating bats. *J. Acoust. Soc. Am.* 51, 2063–2065. doi: 10.1121/1.1913069

Peng, B., Jin, L., and Shang, M. (2021). Multi-robot competitive tracking based on k-WTA neural network with one single neuron. *Neurocomputing* 460, 1–8. doi: 10.1016/j.neucom.2021.07.020

Pérez-Carabaza, S., Scherer, J., Rinner, B., López-Orozco, J. A., and Besada-Portas, E. (2019). UAV trajectory optimization for Minimum Time Search with communication constraints and collision avoidance. *Eng. Appl. Artif. Intell.* 85, 357–371. doi: 10.1016/j.engappai.2019.06.002

Qi, Y., Jin, L., Luo, X., Shi, Y., and Liu, M. (2021). Robust k-WTA network generation, analysis, and applications to multiagent coordination. *IEEE Trans. Cybern.* 1–13. doi: 10.1109/TCYB.2021.3079457

Rostami, S. M. H., Sangaiah, A. K., Wang, J., and Liu, X. (2019). Obstacle avoidance of mobile robots using modified artificial potential field algorithm. *EURASIP J. Wireless Commun. Netw.* 2019, 1–19. doi: 10.1186/s13638-019-1396-2

Sändig, S., Schnitzler, H. U., and Denzinger, A. (2014). Echolocation behaviour of the big brown bat (*Eptesicus fuscus*) in an obstacle avoidance task of increasing difficulty. *J. Exp. Biol.* 217, 2876–2884. doi: 10.1242/jeb.099614

Shin, Y., and Kim, E. (2021). Hybrid path planning using positioning risk and artificial potential fields. *Aerosp. Sci. Technol.* 112:106640. doi: 10.1016/j.ast.2021.106640

Simmons, J. A. (1973). The resolution of target range by echolocating bats. *J. Acoust. Soc. Am.* 54, 157–173. doi: 10.1121/1.1913559

Simmons, R. (1996). "The curvature-velocity method for local obstacle avoidance," in *Proceedings of IEEE International Conference on Robotics and Automation, Vol. 4* (Minneapolis, MN: IEEE), 3375–3382.

Surlykke, A., Ghose, K., and Moss, C. F. (2009). Acoustic scanning of natural scenes by echolocation in the big brown bat, *Eptesicus fuscus*. *J. Exp. Biol.* 212, 1011–1020. doi: 10.1242/jeb.024620

Ulrich, I., and Borenstein, J. (1998). "VFH+: reliable obstacle avoidance for fast mobile robots," in *Proceedings IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), Vol. 2* (Leuven: IEEE), 1572–1577.

Wu, M., Dai, S. L., and Yang, C. (2020). Mixed reality enhanced user interactive path planning for omnidirectional mobile robot. *Appl. Sci.* 10:1135. doi: 10.3390/app10031135

Yasin, J. N., Mohamed, S. A., Haghbayan, M. H., Heikkonen, J., Tenhunen, H., and Plosila, J. (2020). Unmanned aerial vehicles (uavs): collision avoidance systems and approaches. *IEEE Access* 8, 105139–105155. doi: 10.1109/ACCESS.2020.3000064

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

ORIGINAL RESEARCH
published: 24 June 2022
doi: 10.3389/fnbot.2022.864380

Model-Based and Model-Free Replay Mechanisms for Reinforcement Learning in Neurorobotics

Elisa Massi, Jeanne Barthélemy, Juliane Mailly, Rémi Dromnelle, Julien Canitrot, Esther Poniatowski, Benoît Girard† and Mehdi Khamassi†*

Institut des Systèmes Intelligents et de Robotique, UMR 7222, CNRS, Sorbonne Université, Paris, France

Experience replay is widely used in AI to bootstrap reinforcement learning (RL) by enabling an agent to remember and reuse past experiences. Classical techniques include shuffled-, reversed-ordered- and prioritized-memory buffers, which have different properties and advantages depending on the nature of the data and problem. Interestingly, recent computational neuroscience work has shown that these techniques are relevant to model hippocampal reactivations recorded during rodent navigation. Nevertheless, the brain mechanisms for orchestrating hippocampal replay are still unclear. In this paper, we present recent neurorobotics research aiming to endow a navigating robot with a neuro-inspired RL architecture (including different learning strategies, such as model-based (MB) and model-free (MF), and different replay techniques). We illustrate through a series of numerical simulations how the specificities of robotic experimentation (e.g., autonomous state decomposition by the robot, noisy perception, state transition uncertainty, non-stationarity) can shed new lights on which replay techniques turn out to be more efficient in different situations. Finally, we close the loop by raising new hypotheses for neuroscience from such robotic models of hippocampal replay.

Keywords: hippocampal replay, reinforcement learning, neurorobotics, model-based, model-free

Edited by:
*Jun Tani,
Okinawa Institute of Science and Technology Graduate University, Japan*

Reviewed by:
*Paul F. M. J. Verschure,
Radboud University Nijmegen, Netherlands
Ming Wang,
Shandong Jianzhu University, China*

****Correspondence:***
*Elisa Massi
massi@isir.upmc.fr*

†These authors share senior authorship

Received: *28 January 2022*
Accepted: *05 May 2022*
Published: *24 June 2022*

Citation:
Massi E, Barthélemy J, Mailly J, Dromnelle R, Canitrot J, Poniatowski E, Girard B and Khamassi M (2022) Model-Based and Model-Free Replay Mechanisms for Reinforcement Learning in Neurorobotics. Front. Neurorobot. 16:864380. doi: 10.3389/fnbot.2022.864380

1. INTRODUCTION

For a reinforcement learning (RL) agent (Lin, 1992; Sutton and Barto, 1998), experience replay consists of storing in (episodic) memory a buffer containing a series of observations (i.e., a quadruplet composed of the previous state, the action, the new state, and the reward) and periodically replaying elements from this buffer to bootstrap learning during offline phases (i.e., between phases where the agent acts and samples new observations in the real-world) (Fedus et al., 2020).

Several important parameters have an impact on the performance of RL agents with experience replay, such as the size of the memory buffer (Zhang and Sutton, 2017), the relative time spent learning from replay vs. the time spent collecting new observations in the world (Fedus et al., 2020), or whether to shuffle the memory buffer and uniformly sample elements from it or prioritize elements as a function of their associated level of surprise (e.g., the absolute reward prediction error associated to a given quadruplet observed from the environment) (Moore and Atkeson, 1993; Peng and Williams, 1993; Schaul et al., 2015).

To our knowledge, these replay techniques have their origin in the 1990s, when Long-Ji Lin at Carnegie Mellon University proposed solutions to enable RL reactive agents [i.e., model-free (MF) agents such as Q-learners Watkins, 1989] to bootstrap their learning process in large dynamic (non-stationary) discrete simulation environments (Lin, 1992). One of the investigated solutions was to use the Dyna-Q architecture (Sutton, 1990) to learn action models and use these models to sample hypothetical actions. Another tested solution consisted of storing the agent's experience in a memory buffer and replaying it to bootstrap learning. Interestingly, one of the main results was that the best performance was obtained by reversing the order of the replay buffer, what we will call *backward replay* (i.e., replaying first the most recent observation, then the second-to-last one, and so on until the oldest observation). This is because each time the buffer contains a rewarding observation, replay leads to increasing the value of the action performed in the previous state, followed by replaying precisely that previous state at the next iteration (because the buffer is in reverse order), and thus increasing the value of the preceding action, and so on. As a consequence, single processing of the memory buffer results in reward value propagation from rewarding states along the whole sequence of actions that the agents had experienced to get the reward.

In parallel, other researchers further investigated the efficiency of model-based (MB) techniques to sample hypothetical actions rather than replaying experienced actions from a memory buffer. One example is *prioritized sweeping* and consists of replacing uniform model sampling with a prioritization that depends on the absolute value of the reward prediction error (Moore and Atkeson, 1993; Peng and Williams, 1993). While MB methods can be conceived as ways of planning, thus different from MF learning, they can nevertheless be seen as an alternative way to perform offline Q-value updates. Even further, there is a mathematical equivalence between the sequence of Q-values obtained with MB updates and with MF methods with replay (van Seijen and Sutton, 2015). This is why throughout this paper, we will discuss both *model-based* and *model-free replay*, in the sense that they represent alternative offline reactivation mechanisms to update action values. We will refer to model sampling as *Simulation Reactivations* (SimR) and sampling from a memory buffer as *Memory Reactivations* (MemR).

Strikingly, neuroscience research has found that the mammalian brain also seems to perform some sort of experience-dependent reactivations of neural activity, in particular, in a part of the brain called the *hippocampus* (Wilson and McNaughton, 1994). These reactivations occur either when an animal is sleeping (Ji and Wilson, 2007) or during moments of quiet wakefulness between trials of the task (Karlsson and Frank, 2009). Most importantly, these reactivations play an instrumental role in learning and memory consolidation, since blocking these neural reactivations leads to impaired learning performance (Girardeau et al., 2009; Ego-Stengel and Wilson, 2010; Jadhav et al., 2012), while new memories can be created by stimulating reward circuits during these reactivations (De Lavilléon et al., 2015).

The computational neuroscience literature has recently compared the different replay techniques from machine learning with the properties of hippocampal replay recorded experimentally (Pezzulo et al., 2017; Cazé et al., 2018; Mattar and Daw, 2018; Khamassi and Girard, 2020). Interestingly, the reactivation of a sequence of states experienced by the animal during the task sometimes occurs in the same *forward* order and sometimes in *backward* order (Foster and Wilson, 2006; Diba and Buzsáki, 2007). Nevertheless, a large part of hippocampal reactivations occur in apparent random order, and the underlying computational principle remains to be explained (see for instance the proposal of Aubin et al., 2018). Moreover, computational investigations recently found that prioritized sweeping can also explain some properties of hippocampal reactivations (Cazé et al., 2018; Mattar and Daw, 2018). But it is not yet clear whether a single unified computational principle can explain hippocampal replay or whether the brain alternates between different types of replay (backward, shuffled, prioritized/MF vs. MB) in different situations (sleep vs. quiet wakefulness, depending on the difficulty of the task, the level of noise/uncertainty).

Thus, a new field of neurorobotics research is currently dedicated to integrating offline reactivations in the RL processes to improve and speed them up. As mentioned above, this focus on offline reactivations is both inspired by the machine learning techniques created in the 90s and now commonly used in DeepRL and by the neuroscience results on hippocampal reactivations and the probable cohabitation of MB and MF RL systems in the brain. With robotic applications as an aim, these contributions need to bridge the gap between perfectly controlled discrete state simulations and real embodied robotics experiments in continuous environments. The goal of this research is to understand which replay techniques give the best learning performance in different situations (constrained corridor-based vs. open maze environments; non-stationary goal locations and maze configurations) and whether robotic tests lead to different conclusions than simple perfectly controlled simulations (physical vs. abstract simulations, autonomous state decomposition by the robot, noisy perception). For instance, a recent neural network-based simulation of a rat maze task highlighted that shuffled experience replay was required to break the data temporal correlations to be able to learn a neural internal world model (Aubin et al., 2018). Importantly, while neurorobotics research during the last 20 years had already studied hippocampus models for robot navigation (Arleo and Gerstner, 2000; Fleischer et al., 2007; Dollé et al., 2008; Milford and Wyeth, 2010; Caluwaerts et al., 2012; Jauffret et al., 2015), to our knowledge, the impact of different types of replay on the performance of these models has only recently started to be investigated.

In this paper, we illustrate this line of research by presenting a series of numerical simulations of laboratory mazes (used to study rat navigation in neuroscience) as benchmark tasks for robotic learning. These simulations are presented in order of increasing complexity toward real-world robotic experiments. At each step of this presentation, we simulate and compare different replay techniques in either MF or MB RL agents. We discuss the properties of these simulations, how they contribute to improving learning in robots, and how they can also help generate predictions for neuroscience.

2. SIMULATION OF INDIVIDUAL REPLAY STRATEGIES IN A PREDEFINED DISCRETE STATE SPACE

In this section, we present a series of numerical simulations in a simple deterministic maze task with predefined state decomposition. The task mimics the multiple T-maze of Gupta et al. (2010), where rats have to follow constrained corridors and make binary decisions (go left or go right) at specific T-like decision points (**Figure 1A**). This will enable us to first illustrate the properties of different replay methods in the same conditions as the perfectly controlled simulations usually performed in computational neuroscience work. Then in the next sections, we will study what happens in more open mazes where moreover the robot will autonomously build its state decomposition.

The work presented in this section contains two main differences from our previous computational neuroscience simulations of the multiple T-maze task (Cazé et al., 2018; Khamassi and Girard, 2020)[1]: (1) in previous work, following experience replay techniques in machine learning, we had allowed the agent to perform a series of replay iterations after each action; here, because it would be energy- and time-consuming for a robot to stop after each action, we allow the simulated robot to perform replay only at the end of the trial, while it is waiting for the next trial at the departure state; (2) we had simulated a version of *MB prioritized sweeping* where the memory buffer contained one element per state; here, we test whether it is also efficient to have an element for each (state,action) couple, thus filling the memory buffer with multiple elements for the same state (as long as they represent different actions).

2.1. Methods

We simulate the multiple T-maze task as a Markov decision problem (MDP), where an agent visits discrete states $s \in \mathcal{S}$, using a finite set of discrete actions $a \in \mathcal{A}$. States represent here unique locations in space, equally spaced on a square grid (**Figure 1A**), a piece of information expected to be provided by place cell activity in the hippocampus (O'Keefe and Dostrovsky, 1971). The actions allowed the agent to represent moves in the four cardinal directions: north, south, east, and west. During the first 100 trials, the reward will always be located on the left arm. Then during the next 100 trials, the reward will be on the right arm and the agent will have to adapt its decisions accordingly.

Here, we simulate three model-free RL algorithms and one MB one: MF without replay, MF with backward replay, MF with shuffled replay, and MB prioritized sweeping (**Table 1**).

For each Markovian state-action couple (s, a) in the environment, MF-RL agents use Q-learning (Watkins, 1989) to learn the Q-value of performing action a from state s, as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

[1]The updated code for these simulations is available at https://github.com/MehdiKhamassi/RLwithReplay.

Where $R(s, a)$ is the reward obtained from the environment when performing (s, a), and s' is the arrival state after executing action a in state s.

At the next timestep, deciding which action to perform is computed by drawing the next action a from a probability distribution given by the softmax Boltzmann function applied to the Q values:

$$P(a|s) = \frac{e^{\beta Q(s,a)}}{\sum_{i \in \mathcal{A}} e^{\beta Q(s,i)}} \quad (2)$$

With $\mathcal{A}$ being the set of all the possible actions from state s and β being the inverse temperature parameter that regulates the compromise between exploration and exploitation: the closer to zero, the more the differences between the Q-values will be attenuated, and thus the more the selection will be uniform (hence exploratory); conversely, large values (that can go up to infinity) will enhance the contrast between the Q-values and will thus favor exploitation of the largest one.

In *MF-RL backward replay* and *MF-RL shuffled replay* and for all the other RL replay algorithms tested in this section and the next one (Section 3), we enable the agent at each timestep to store in a memory buffer the quadruplet describing the current observation: the previous state s from which the agent performed action a, the resulting state s' and the scalar reward r obtained from the environment (1 when the rewarding state has been reached, 0 elsewhere). This memory buffer progressively increases in size, timestep after timestep, but is limited by the maximal size N (N being chosen to correspond to the number of states in the environment, see **Table 1**). When the maximal size has been reached, adding a new element to the buffer is accompanied by throwing away the oldest element in it.

When the agent has finished the current trial and reaches the departure state again, a replay phase is initiated where at each replay iteration one element from the buffer is processed and the corresponding Q-value is updated following Equation 1. This is repeated until the sum of variations of Q-values over a window of N replay iterations is below a certain replay threshold ϵ, which indicates that the Q-values have converged and do not require to be updated anymore.

In the *MF-RL backward replay* algorithm (Lin, 1992), at the beginning of a new replay phase, we simply reverse the order of elements in the buffer and then start to perform replay iterations following the procedure explained above. In the *MF-RL shuffled replay* algorithm, we simply shuffle the elements of the buffer before starting the replay phase.

We also test an MB algorithm where the learning process aims at building a world model, i.e., a model of how the perceived world changes when actions are taken. This model is conventionally composed of a transition function and a reward function. The transition function $T(s, a, s')$ represents the probability of observing s' next if action a is taken while in state s. In the present discrete case, it is built by storing the number of times each (s, a, s') triplet was encountered and by dividing by the number of times (s, a) experienced, as shown in the equation below:

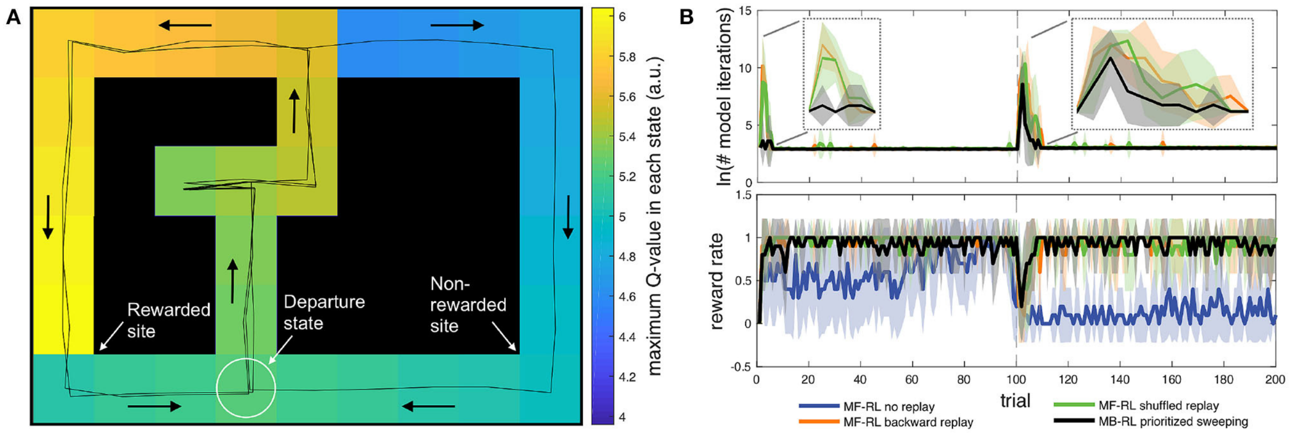


FIGURE 1 | (A) Discrete state-space simulations in the multiple T maze task (Cazé et al., 2018; Khamassi and Girard, 2020). The reward is on the left side for 100 trials and then shifted to the right side for the next 100 trials. In the present simulations, replay is only allowed in the departure state, before starting the next trial. Despite this constraint, the figure shows that after only 3 trials (2 correct / 1 error), the MF-RL algorithm with backward replay has already learned a full gradient of Q-values across the maze. **(B)** Comparison of the performance (reward rate) and computation time (Napierian logarithm of the number of iterations during replay phases) for 4 different algorithms. The thick lines represent the average, and the area around represents the mean square error. The figure illustrates that MF-RL without replay requires 60–70 trials to reach optimal performance and does not manage to adapt to the change in reward location within only 100 trials. All the other algorithms perform similarly in terms of reward rate: fast increase in performance; brief drop in performance after the change in reward location; fast re-increase of performance afterward. These algorithms mainly differ in the required duration of the replay phases: MF-RL with random replay and MF-RL with backward replay both show a strong peak in the number of replay iterations after the change in goal location. The state-based version of MB-RL prioritized sweeping shows a smaller peak.

TABLE 1 | Algorithm parameters used to generate the results in this section.

	MF no replay	MF backward replay	MF shuffled replay	MB prioritized sweeping
α	0.2	0.2	0.2	-
γ	0.99	0.99	0.99	0.99
β	3	3	3	3
ϵ	-	0.001	0.001	0.001
N	-	54	54	54

They have been taken from Cazé et al. (2018) without retuning. α is the model-free (MF) learning rate. γ is the discount factor. β is the inverse temperature in the softmax for decision-making (Equation 2). ϵ is the threshold for Q-values convergence during replay. N is the maximal size of the episodic memory buffer.

$$T(s,a,s') = \frac{V_N(s,a,s')}{V_N(s,a)} \tag{3}$$

where $V_N(s,a)$ stands for the number of visits of state s when action a is then chosen and $V_N(s,a,s')$ is the number of transitions from state s to state s', having performed action a. The reward function $R(s,a,s')$ represents the average reward signal experienced when effectively performing the (s,a,s') transition. For the *MB-RL prioritized sweeping* algorithm that we simulate here (Moore and Atkeson, 1993; Peng and Williams, 1993), we add to each element in the memory buffer the absolute reward prediction error Δ measured when experiencing (s,a,s',r) in world. This Δ can also be seen as representing the magnitude of change in $Q(s,a)$ which resulted from this observation. The memory buffer is sorted in decreasing order of Δ, thus giving a high priority to be replayed to elements representing surprising events in the world that resulted in important revisions of Q-values. In fact, Mattar and Daw (2018) have formally shown that deriving the *Expected Value of (Bellman) Backup* (in other words an expected value of doing a replay) leads to maximizing a *gain* term which is higher for transitions that have been associated with larger reward prediction errors (hence larger surprise) when the agent was experiencing the real world.

During the replay phase of *MB-RL prioritized sweeping*, we start by considering the first element (s,a) of the buffer with the highest Δ. We use the world model learned by the agent to estimate the virtual reward r and arrival state s', and then apply one iteration of the *Value Iteration* algorithm (Sutton and Barto, 1998) to update the Q-value of (s,a), where k is all the possible actions starting from the arriving state s':

$$Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} T(s,a,s') max_{k \in \mathcal{A}} Q(s',k) \tag{4}$$

From Equation 4, we can compute the new Δ for the couple (s,a) and reinsert it within the memory buffer with Δ as the new priority level. Finally, we use the world-model to find all possible predecessors of (s,a), i.e., couples (s'',a''), which according to the model enable the agent to reach state s. Because the predecessors of a given state s can be difficult to determine in a stochastic world, Moore and Atkeson (1993) propose to consider as predecessors all the states s'' which have, at least once in the history of the agent in the current task, performed a one-step transition $s'' \rightarrow s$. The priority associated to a predecessor s'' can thus be the corresponding absolute prediction error Δ_{pred} and determined in which position it will be inserted in the memory buffer, as introduced by Peng and Williams

(1993). The replay phase then continues by processing the next element in the buffer with the highest priority level, and so on, until one of the stop conditions described above is met. For the sake of terminological clarification, what we call here a replay phase for an MB algorithm corresponds to an inference phase. This is because *MB-RL prioritized sweeping* does not replay memorized past experience, but rather generates SimR through model sampling combined with the value iteration algorithm described above. Thus, to transpose from MF to MB, the replay phase stop conditions described above, the size of the replay budget *N* (which could also be called an inference budget in the case of MB) represents here a maximum number of iterations that can be inserted in the prioritized memory buffer and replayed through the value iteration algorithm.

2.2. Results

With the two changes that we made here compared (Cazé et al., 2018; Khamassi and Girard, 2020) (i.e., (1) only allowing the simulated robot to do replay at the end of the trial when reaching the departure state and (2) storing distinct (state, action) couple in the memory buffer for *MB-RL prioritized sweeping* rather than a single element per state), we found consistent performance results and only a difference in terms of a reduced computational cost during replay phases, which we describe below.

Figure 1B shows that the three algorithms with replay (*i.e., MF-RL backward replay, MF-RL shuffled replay*, and *MB-RL prioritized sweeping*) quickly reached the optimal reward rate of 1 at the beginning of learning and then experienced only a brief drop in reward rate after the change in reward location at trial #100. In contrast, *MF-RL without replay* took longer to reach the optimal rate (approx. 60 trials) and then barely managed to re-increase its reward rate within 100 trials after the change in reward location. So, the first conclusion is that any replay technique is equally useful in enabling fast learning in such a simple maze task with predefined state decomposition.

The second interesting observation has to do with the transient and nearly discrete increases in replay time that are produced in responses to task changes (**Figure 1B**). All replay techniques enable the agent to avoid spending time performing replay during the majority of the task. They moreover show a sharp increase in replay time after a change in reward location. Importantly, this property was also true in our previous work where replay was not restricted to the end of the trial but rather allowed in any state of the task (Cazé et al., 2018). Thus, it is interesting to note that such a way to generate replay events is not only compatible with neurobiological data (Cazé et al., 2018; Mattar and Daw, 2018) but also shows properties that could be useful for autonomous robots: bursts of replay could be used by the robot as a way to automatically detect new task conditions (but here the robot does not need to explicitly label these events; it just needs to adapt and maximize reward). The rest of the time, the agent starts each new trial without pausing, as if not showing any hesitation, similar to what is classically observed in well-trained rats in similar tasks (Gupta et al., 2010).

In addition, it is interesting to compare the duration of replay phases between the different replay techniques. While there is no difference in the average number of replay iterations after the change in reward location at trial #100 (**Figure 1B**), *MB-RL prioritized sweeping* performs drastically fewer replay iterations than *MF-RL backward replay* and *MF-RL shuffled replay* during the initial learning phase (first 5-10 trials of the task). Now that we restricted these algorithms to perform replay only at the end of each trial, rather than after each action during the trial, *MB-RL prioritized sweeping* performs even fewer replay iterations than what we previously obtained in the same task (Cazé et al., 2018), without affecting its reward rate. The new proposal to restrict replay to the inter-trial interval thus seems promising for real robots. In Dromnelle et al. (2020b) (where we had not implemented any replay mechanism yet), the robot indeed took a few seconds after each trial to go back to the departure state. This short moment seems ideal to let the algorithm perform a replay without affecting the performance of the robot during the trial.

In the next section, we keep these principles and compare the same replay algorithms in a more open environment where the robot autonomously learns to decompose the task into discrete states, to verify that these algorithms still perform well under these more realistic conditions.

3. SIMULATION OF INDIVIDUAL REPLAY STRATEGIES WITH AN AUTONOMOUSLY LEARNED STATE DECOMPOSITION

The neural activity of hippocampal place cells is often observed as showing transients and increases after surprising events (Valenti et al., 2018). During maze navigation, surprising events mostly occur at locations in the environment that are associated with positive or negative outcomes. From these locations, reverse replay, in particular, could reinforce spatial learning by occurring during awake periods, after the spatial experiences (Foster and Wilson, 2006). They can potentially reinforce the surprising experience by propagating the outcome of the event to states that have been encountered by the animal on its way to the reward or punishment site. Moreover, rewarding states are also very likely to initiate replay activity in the hippocampus to enhance the memory consolidation of novel information (Michon et al., 2019). During these events, the reactivation of the hippocampus neural activity is thought to be initiated by rewarding outcomes to bind this positive unexpected experience to the events that preceded it (Singer and Frank, 2009).

To study these and others phenomena related to spatial navigation learning in rodents, one of the first and most relevant experimental protocols is the *Morris Water Maze* (Morris, 1981). In this work, rats were introduced to a circular pool with opaque water and were removed from the pool only after reaching a hidden platform, located just below the water surface. Even though the rats could not see the platform, they were still able to spatially localize it. This was found even in cases where their starting point changed within the pool, thus indicating a robust spatial memory.

In this section, the same MF-RL and MB-RL replay strategies (MemR and SimR, Sections 1, 2) are tested in a more realistic robotic set-up, where the discretization of the environment in multiple Markovian states is autonomously performed by the robot.[2] Similarly to the experiment in Section 2 and to what has been experimentally observed by Foster and Wilson (2006), the replay phase takes place once the agent has reached the reward state to enable offline learning of Q-values, as previously done by Mattar and Daw (2018). Neurobiologically, even though *Vicarious Trial and Error (VTE)* plays an important role in animals' reasoning and decision-making (Tolman, 1939; Redish, 2016), it usually happens in uncertain moments, such as at beginning of the experiment, at the decision points or surprising spots (Cazé et al., 2018; Khamassi and Girard, 2020) and can also be unconsciously constrained by the attempt to limit the opportunity cost (Keramati et al., 2011).

This aspect is particularly crucial for the robotic experiment because it allows the agent to spend the Inter Trial Interval (ITI) updating the Q-table, based on a replay of its past experience. Usually, this time interval does not require expensive computations for the robot, since it does not need to take any decision on its way back to the starting position, and by replaying past experience, the learning speed could be enhanced without losing important experimental time.

The addressed research question is whether MF-RL or MB-RL replay strategies could enhance spatial learning for artificial agents and robots. We found it interesting to first test our proposed algorithm in a simulated version of an experimental task (Morris, 1981) and eventually investigate if there were any differences between replaying reverse sequences of actions, random transitions, or the most surprising transitions, similarly to what has been done in Section 2.

Like in the previous section, the presented simulated experiment investigates the role of diverse replay strategies relative to a changing reward condition. Moreover, the aim is also to investigate whether replays are relevant when transitions between the states of the task are stochastic. These simulations thus bring us to more realistic robotic experiments, in stochastic and dynamical environments.

3.1. Materials and Methods

To study the implications of offline learning in spatial navigation, from rodents' behavior to robotics, we have first investigated the role of two MF- and one MB-RL replay techniques (as in Section 2) in a circular maze, consistent with the original Morris water maze task (Morris, 1981) in terms of environment/robot size ratio. The learning performances of the analyzed replay techniques are discussed in two main conditions:

- A deterministic version of the task, where an action a performed in a state s will always lead the robot to the same arrival state s' with probability 1.
- A stochastic version of the task, where performing action a in state s is associated with non-null probabilities of arriving in more than one state.

[2]The code for these simulations is available at https://github.com/esther-poniatowski/Massi2022.

TABLE 2 | Algorithm parameters are used to generate the results in this section.

	No replay	MF backward replay	MF shuffled replay	MB prioritized sweeping
α	0.8	0.8	0.8	0.8
γ	0.9	0.9	0.9	0.9
β	15	15	15	15
ϵ	-	0.001	0.001	0.001
N	-	90	90	90

*α is the learning rate, optimized as shown in **Figure 4** and Equation 6, and γ is the discount factor. β is the inverse temperature in the softmax function for decision-making (Equation 2), and its values were found by optimizing both the convergence time and the performance of the tested algorithms. N is the maximal length of the episodic memory buffer. This value was selected to replay the entire real experience during the first trials of the experiment and to replay experiences from several past trials later in the simulation. Finally, ϵ is the convergence threshold as for Sections 2 and Cazé et al. (2018).*

3.1.1. Learning Algorithm and Replay

As in the previous series of simulations (Section 2), the simulated agent is learning using either classical *MF-RL Q-learning* (Watkins, 1989) (Equation 1) or *MB-RL prioritized sweeping learning* (Moore and Atkeson, 1993; Peng and Williams, 1993). The values of their parameters (learning rate α and the discount factor γ) are shown in **Table 2**.

The first implementation of offline learning techniques that we tested is the *MF backward replay*. Similar to the double T-maze experiment in Section 2, the offline learning phase happens once the agent has reached the reward state, which indicates the end of a trial. During a trial, the Q-values $Q(s, a)$ of the state-action couple (s, a) are updated with Equation 1 and once the rewarding state has been reached, they are updated again in reverse order, starting from the reward state. These backward sequences can be up to N updates long if the agent has gained enough past experience and stored it in its memory buffer. The reverse sequences are then replayed until the sum of variations of Q-values over the last replay repetition is below a certain replay threshold ϵ (**Table 2**). Given the size of the environment (36 states), these N long backward replay sequences can also involve experiences that happened during the previous trials of the same agent (i.e., during the previous attempts to get to the reward). In this way, the robot can transfer the acquired knowledge through different trials and learn more efficiently.

The second replay strategy that has been tested is *MF shuffled replay*; in this case, in the ITI, the internal values $Q(s, a)$ are randomly ordered and then updated by Equation 1. As for the *MF backward replay*, the memory buffer, that is accessible to initiate the reactivations, keeps in memory the latest N transitions (**Table 2**). Also, in this case, the agent can benefit from the experience acquired during the latest trials and learn to extract more general and useful knowledge from its recent and uncorrelated past actions (because of shuffling). The ITI replay phase lasts until the convergence of the sum of the Q-values under an ϵ value given in **Table 2**.

As for Section 2, we compared the learning performance of the above-explained MF replay strategies to an MB prioritized sweeping algorithm (Moore and Atkeson, 1993; Peng and Williams, 1993). The implementation of the latter is the same as described in Section 2.1, and the convergence criterion is reached when the prioritized replay buffer, which can be maximum N transitions long, is empty.

3.1.2. The Experimental Set-Up and Implementation

The simulated experimental set-up intends to replicate a Morris water maze task: the agent is introduced in a new circular environment, and it has to learn how to reach a particular location associated with a positive reward (Morris, 1981). In our set-up, the agent is a Turtlebot3 Burger, simulated with the Robot Operating System (ROS) middleware and the Gazebo simulation environment (Quigley et al., 2009). The water maze is represented as an empty circular arena surrounded by high walls (**Figure 2B**).

The robot discovers and defines the different discretized areas in the maze by autonomously navigating within the environment. Despite the odometry and the laser sensor being installed on the robotic device, the acquired space representation is allocentric. This is an emergent property of the automatic clustering process when applied to robot sensor data in a task where the robot can only move in a horizontal plane, as found in previous neurorobotics work (Caluwaerts et al., 2012). The robot, in fact, explores by selecting between 8 directions of motion that are defined in the global reference frame of the environment, and its current position and orientation are also elaborated in the maze reference frame. This allocentric description of the robot movements and the states of the maze is possible thanks to a re-mapping of the relative position of the robotic agent and the discretized states to the reference coordinate system of the map. This is possible thanks to the 360 Laser Distance Sensor of the robotic platform, combined with the use of a classical SLAM technique. Note that such an allocentric space representation is not only compatible with neurophysiology (hippocampal place cell activity) but can also be combined with egocentric representations to account for a variety of experimentally observed animal behaviors during navigation tasks (Khamassi and Humphries, 2012). The discrete MDP, presented in **Figure 2A**, is obtained thanks to a Rao-Blackwellized particle filter that builds grid maps from laser range data (Grisetti et al., 2007). The simulated implementation of this Simultaneous Location and Mapping Algorithm (SLAM) on ROS Gazebo is called *GMapping*.

This state decomposition process makes the robot able to immediately create new states if necessary, but in our work, the aim was to create the finest and most robust possible discretization of the maze to be then employed in all the simulation experiments where we tested the different replay strategies. As observed by Khamassi (2007), Chaudhuri et al. (2019), and Benchenane et al. (2010), rats could re-explore the whole maze every day before doing a learning task and that could reflect their need to rapidly acquire and stabilize a state representation before starting an extra learning process.

For these reasons, the robot performs a long autonomous exploration phase to acquire its state representation before starting the learning phase. During the first 48-min-long exploration in Gazebo, the SLAM algorithm estimates the current robot coordinates, and whenever it is more than 15 cm further away from any existing state, it creates a new state, whose reference position is the current position. This results in a Voronoi partition of the space, composed here of 36 states (**Figure 2A**). This 15 cm state radius was chosen to be similar to the robot footprint of 13,8 x 17,8 x 19,2 (L x W x H, cm). The action space $\mathcal{A}$ instead contains 8 homogeneously distributed directions of motion, defined with respect to the world reference frame (same as for Section 4, **Figure 8A**, top right).

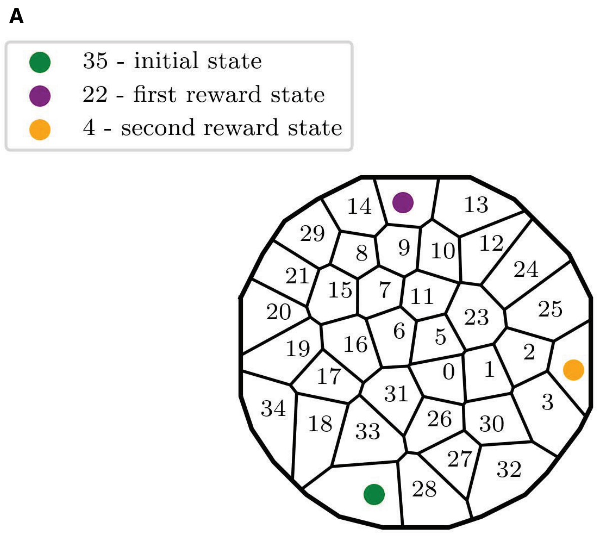
Then, we ran another free exploration of the arena by the simulated Turtlebot3 robot to automatically learn the transition probabilities $p(s'|s,a)$ that can be approximated from randomly executing different actions a in different states s and observing the arrival state s'. This second free exploration phase was chosen to be 5,357-action long, the same duration as for the results that will be presented in Section 4. Lesaint et al. (2014) found that when an agent was progressively learning its transition function during the task, the RL model was better at accounting for rat behavior than a model with a prior given transition function.

In practice, the transition probabilities autonomously learned by the robot during free exploration in Gazebo is stochastic: the same action a performed in the same state s can lead to more than one state with non-null probabilities. For instance, moving north from state #31 alternatively leads to states #0,5,6,16, and even sometimes to state #31 itself when the robot initiated its movement from the bottom part of this state (**Figure 2A**). Such stochasticity results from several properties: (1) because the states autonomously decomposed by the algorithm are not evenly distributed; (2) because the experiments are performed in a simulated physical environment, which includes frictions between the robot's wheels and the floor, and where the robot sometimes moves too close to the walls, thus triggering its obstacle-avoidance process, hence resulting in a different effect of the same action performed without obstacle-avoidance.

The actual level of uncertainty of the stochastic version of the task is displayed in **Figure 10A**, where each state s has an entropy $H_{env}(s)$ computed as in Equation 5, where $\mathcal{A}$ is the set of all the possible actions a from state s, s' are all the possible arrival states from the original state s, and $p(s'|s,a)$ is the probability that the agent arrives in state s' after starting from state s and performing action a:

$$H_{env}(s) = \max_{a \in \mathcal{A}} \sum_{s'} -p(s'|s,a) \log_2 p(s'|s,a) \tag{5}$$

Finally, in order to obtain a deterministic version of the same task from these autonomously learned transition probabilities $p(s'|s,a)$, for each (state,action) couple (s,a), we search for the state s' with the highest probability of arrival (i.e., $s' = \arg\max_{x \in S}[p(x|s,a)]$), and set $p(s'|s,a) = 1$ while setting $p(s''|s,a) = 0$ for all other states $s''(s'' \neq s')$. The deterministic version of the task consists in fact the simplification of the interaction between the robot and the environment, meaning that the trajectories that the robot can cover in the same environment are reduced. To quantify the simplification of the resulting MDP, we have performed an analysis of the trajectories which have



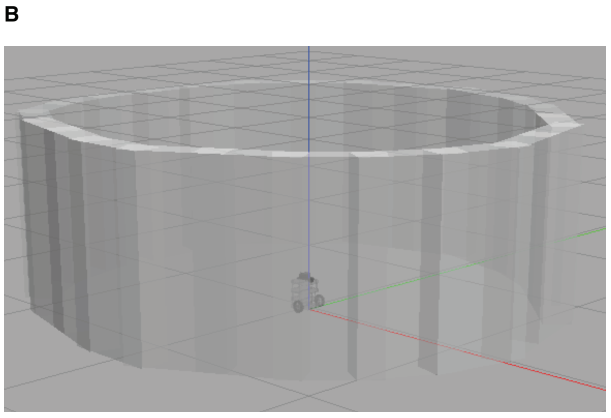


FIGURE 2 | Description of the experimental set-up. **(A)** Map of the discrete states of the maze, identified by the robot during the exploration of Gazebo. The initial state and the two rewarding states are also highlighted. **(B)** The ROS Gazebo simulated Turtlebot 3 in the center of the circular environment.

been taken by the four different algorithms, in the two different environments. We compute the pairwise Fréchet distance of these trajectories to the optimal one, found by following a greedy optimal policy. **Figure 3** shows this analysis during the first half of the experiment when the reward is fixed in state #22. The results from this analysis show that, for all the adopted strategies, in the stochastic environment (**Figure 3B**), the sparsity of the trajectories around the optimal path is generally higher compared to the same deterministic case **Figure 3A**). To assess the difference among these distance distributions, we did a Kruskal-Wallis H-test (Kruskal and Wallis, 1952)), finding them significantly different from each of their corresponding distribution of in the other environment. The conversion of the environment in a deterministic MDP is then intrinsically limiting the level of exploration of the agents, resulting in two very different scenarios. However, it is very important to investigate this transition, given our intent to study the role of RL replay strategies in robotic navigation, from a theoretical to a more realistic robotic outline.

To replicate a non-stationary task similar to the one in the original experiment (Morris, 1981), we changed the reward location from state #22 to state #4 at trial 25, and we tested the learning performances of the agent with four different replay strategies (no replay, MF backward replay, MF shuffled replay, and MB prioritized sweeping) and in two different environments: a deterministic and a stochastic version of the task.

3.2. Results

To assess the real contribution of the tested replay strategies to the learning process of the described spatial navigation task, an unbiased learning rate α_{best} has to be found. Since α_{best} could be different depending on the unpredictability of the MDP which simulates the task (i.e., deterministic or stochastic), we simulated 100 robotic agents performing 50 trials to get to the rewarding states, for a set of uniformly distributed α values between 0 and 1 (**Figure 4**). For each value of α, we looked at the average value $action(\alpha)$ along the trials, with $action(\alpha)$ being the number of actions needed by the robot to get to the rewarding states. These values are computed for both the deterministic (**Figure 4A**), the stochastic worlds, considering the entirety of the experiment, and the minimization of the sum of these two values is used to identify the final α_{best} (**Figure 4B** and **Table 2**) as described in the equation below:

$$\alpha_{best} = \arg\min_{\alpha \in \mathcal{A}} (\overline{action_{deterministic}(\alpha)} + \overline{action_{stochastic}(\alpha)}) \quad (6)$$

where $\mathcal{A}$ is the set of tested α values.

Once identified the most appropriate value for the learning rate α, the following four replay conditions have been tested in the task:

- *MF-RL no replay*
- *MF-RL backward replay*
- *MF-RL shuffled replay*
- *MB-RL prioritized sweeping*

and the other relevant parameters for the experiment are described in **Table 2**.

The main results are shown in **Figure 5**. The four different RL algorithms (no replay, backward replay, shuffled replay, and prioritized sweeping) are compared in terms of the number of model iterations to get to the rewarding state (Napierian logarithm of the first, median, and third percentiles over the behavior of 100 robotic agents). The task changes at trial #25 when the reward switches from state #22 to state #4 (**Figure 2**).

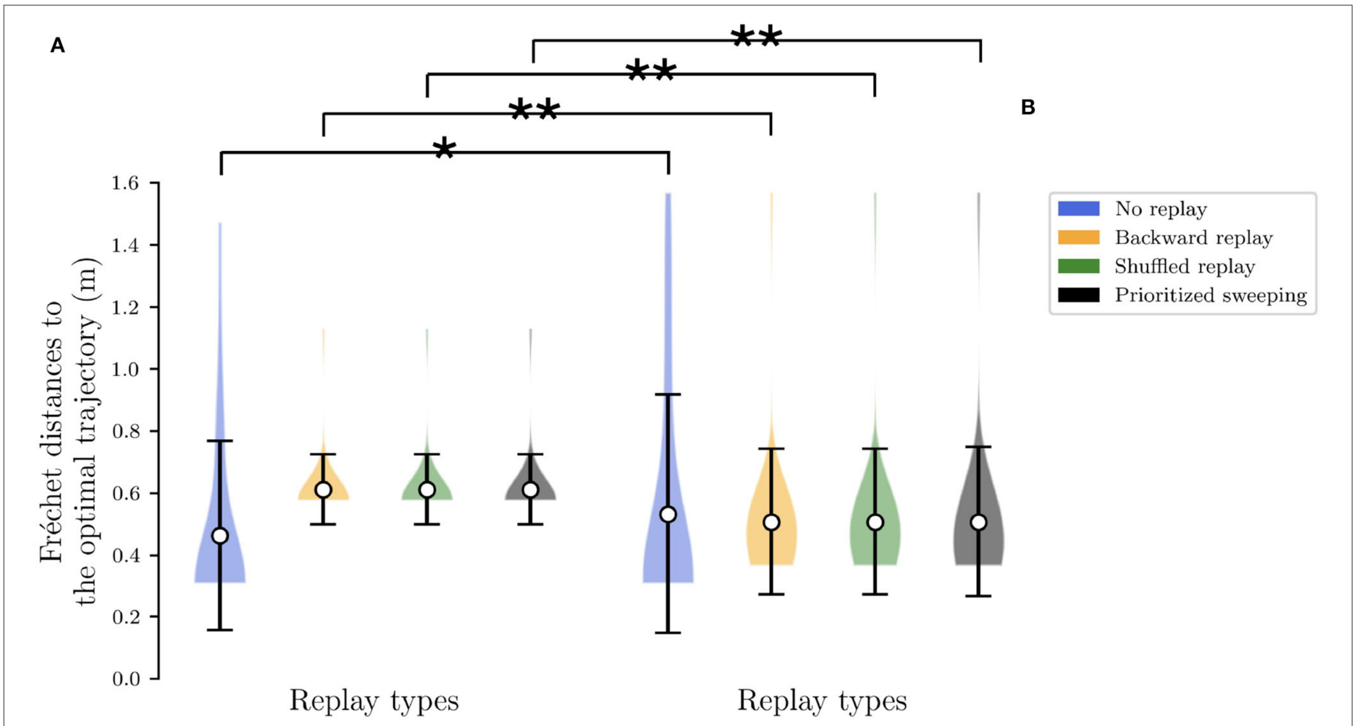


FIGURE 3 | Analysis to investigate the level of the sparsity of the explored trajectories by the agent. The Fréchet distance has been computed for the first half of the simulation. ** Stands for p-value lower than 0.001 and * for p-value lower than 0.05. **(A)** The extension of the Fréchet distance to the optimal trajectory in the deterministic case for all the algorithms. **(B)** The same extension of Fréchet distance in the stochastic environment.

When the task is deterministic (**Figure 5A**), all three RL algorithms with replay learn a short path to the reward significantly faster than the *MF-RL no replay* learner (**Figures 5A,B**, Trials 1-5). The same situation occurs when the reward position is switched at trial #25, assessing the role of RL replays in improving the speed of learning after such a task change (**Figures 5A,B**, Trials 26-30). When the environment is stochastic, the situation is similar and, in particular, the prioritized sweeping algorithm is learning significantly faster than the other replay strategies (**Figure 5B**, Trials 26-30) reflecting the importance of an MB strategy (with MB replay) to faster adapt to dynamical tasks, when the transition model is not deterministic. This suggests that moving toward more complex robotic tasks, MB-RL models of replay may be preferred, since the higher information processing regarding the model of the environment, at the beginning of the task, can save real experimental time, when the robot would need to adapt later in the experiment.

Moreover, the logarithmic scale makes it easier to notice that the no replay agent, even if it is slower at the beginning of the task, can converge to paths that are significantly shorter than the one covered by the other strategies, before the change in reward location (**Figures 5A,B**, Trials 20-25). In the stochastic environment, in particular, the *MB-RL prioritized sweeping algorithm* reinforces the experience of a sub-optimal path, resulting in performance significantly different from the ones obtained from the other two replay strategies (**Figure 5B**, Trials 20–25). This shows that, even if the stochastic environment leads the MF-RL replay strategy to explore more the maze, the *MB-RL prioritized sweeping algorithm*, that can learn the transition model from the beginning of the task, is not subjected to this "push" toward exploration and keeps reinforcing the shortest path previously found.

Instead, in the second convergence phase (Trials 45–50), we highlight the fact that the no replay agent is not showing any more statistically better performances than all the replay algorithms (**Figures 5A,B**, Trials 45–50). In the deterministic case, it is still reaching the shortest path to the reward, but the prioritized sweeping agent is also being significantly better than the *MF-RL shuffled replay* strategy (**Figure 5A** Trials 45-50). On the other hand, in the stochastic case, the *MB-RL prioritized sweeping*'s knowledge of the environment makes it attain performances that are compatible with the ones from the no replay strategy. In this particular case, we can notice that the replay strategies perform differently, with the shuffled replay which performs worse than the other two replay strategies. This re-adaptation phase gives to the agents the opportunity for more exploration, in particular to the replay agents, which have strongly reinforced their previous experienced trajectory to maximize the reward and propagate this knowledge throughout the environment. As already happened in the second learning phase (**Figure 5B**, Trials 26–30), the *MB-RL prioritized sweeping algorithm* significantly exceeds the performance of the other replay algorithms and converges to a shorter path to the reward.

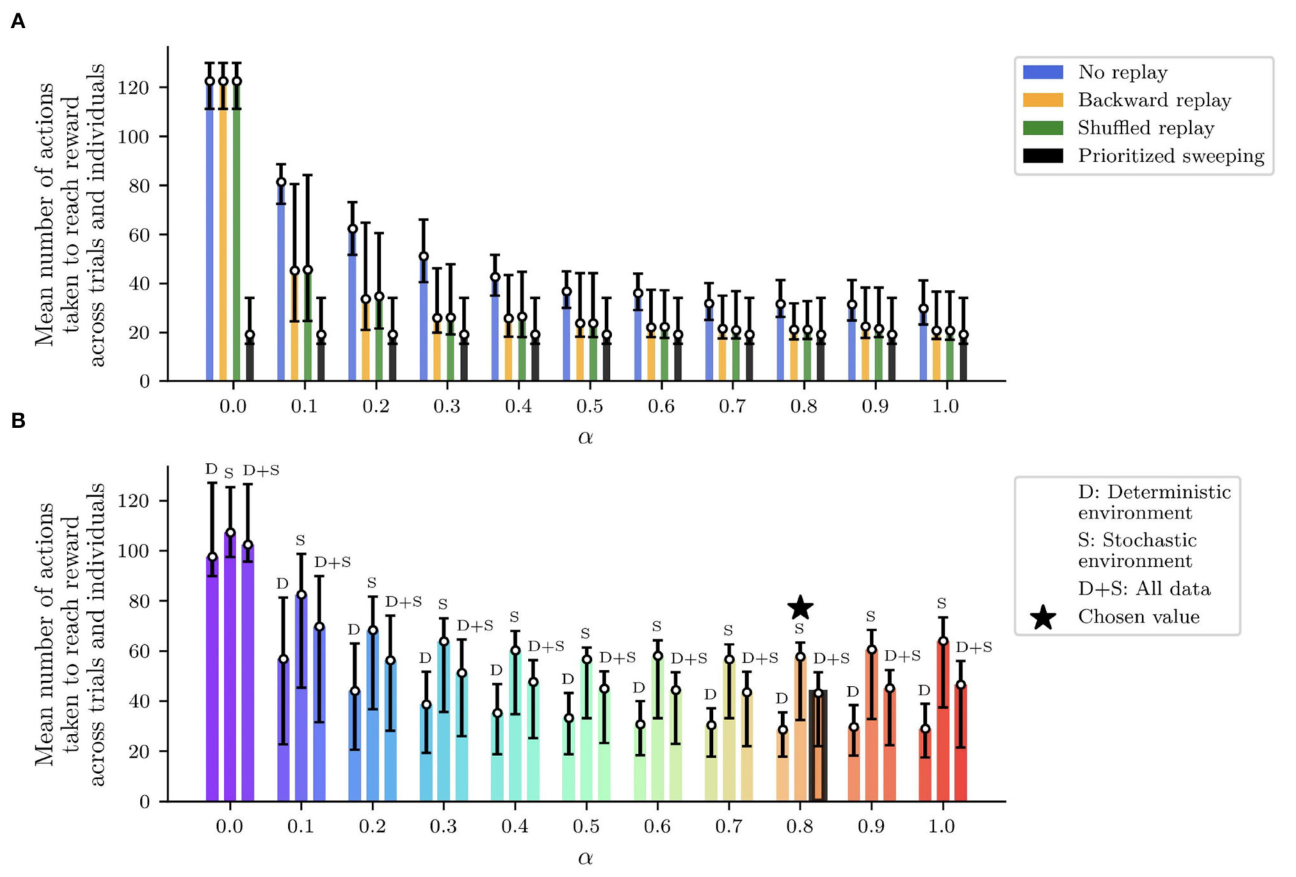


FIGURE 4 | Performed analysis to find out the best learning rate α for all the replay strategies and the two environments (deterministic and stochastic). For different values of α, the figure shows the first, median, and third percentile of the number of actions to get to the reward, over 100 agents completing the simulated experiment over 50 trials. The average minimum number of model iterations to get to the reward is found for α equal to 0.8, and it was used for all the presented experiments (**Table 2**). **(A)** Performances of the tested algorithms across the α values in the deterministic version of the maze. **(B)** Final selection of α considering the mean performances between the deterministic and the stochastic version of the maze.

This gives insights into the need for a more consolidated knowledge of the environment (and so of the interaction of the agent with it) for adaptive tasks. As a consequence, we can predict that animals would need to retrieve knowledge about their experienced and learned model of the world to adapt more efficiently to dynamic circumstances.

Following the results shown in **Figure 5**, we have further investigated the learning and replay dynamics of the proposed strategies. In **Figure 6**, the level of propagation of the Q-values (Equation 1) over the environment is shown for the different tested RL algorithms and for both the deterministic (**Figure 6A**) and the stochastic (**Figure 6B**) environments. The shown learning dynamics are representative of the different strategies since they show the behavior of the individual which is the closest to the median performances of all the 100 individuals for each strategy.

In both cases (**Figures 6A,B**, Trials 1,2 and 25), the presence of replay provides a drastically larger propagation of the Q-values, starting from the first reward state (22). This explains the significantly faster learning performances observed in the algorithms with replay compared to the *MF-RL no replay method*. In both environments, the no replay method is slower to learn, but it explores more in the first trials (Trial 1 and 2) and that leads it to generally find a shorter path to the reward location in the end (Trial 25) compared to the other learning strategies (as shown in **Figure 5A** and, Trials 20–25).

By comparing the two types of environments, we can understand that the level of stochasticity of the MDP leads to a more important exploration of the environment for all the strategies (**Figure 6B**, looking at the explored trajectories and the replayed transitions). This results in a larger propagation of the Q-values in the maze, in particular, in the prioritized sweeping algorithm. As in the deterministic case, the *MB-RL prioritized sweeping* is replaying a broader range of transitions after the first trial compared to the other strategies. With this MB-RL replay strategy, the replay activity is led by the surprise of the experienced events. This results in longer replay phases at specific surprising moments of the task, for instance after the first discovery of the reward location (at trial 1 in **Figure 6**) and after the discovery of the changed

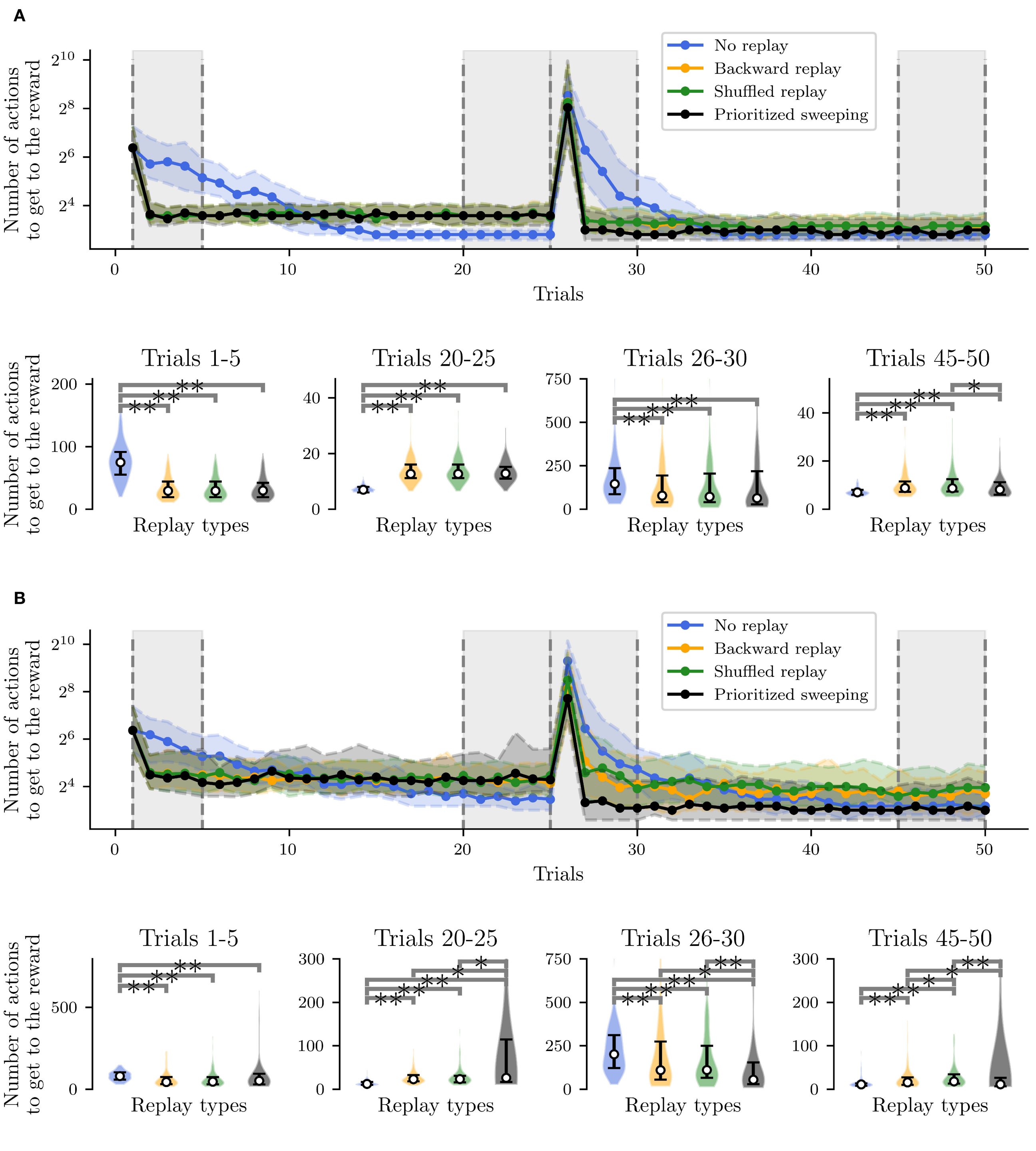


FIGURE 5 | Performances of the simulated robot, learning the non-stationary task, and a *post-hoc* Wilcoxon-Mann-Whitney pairwise comparison test on the relevant trial intervals among the different curves. The *post-hoc* test has been performed following a Kruskal-Wallis *H*-test (Kruskal and Wallis, 1952) to reject the null hypothesis that the population median of all of the algorithms' average performances was equal. ** Stands for *p*-value lower than 0.001 and * for *p*-value lower than 0.05. **(A)** Deterministic environment. **(B)** Stochastic environment.

rewarding state (at trial 26 in **Figure 6**). This happens in both environments also thanks to the implementation of the algorithm which examine also the predecessor of the surprising state (Section 3.1.1) and to the acquired knowledge of the environment (in particular in Trials 26, when the reward position changes).

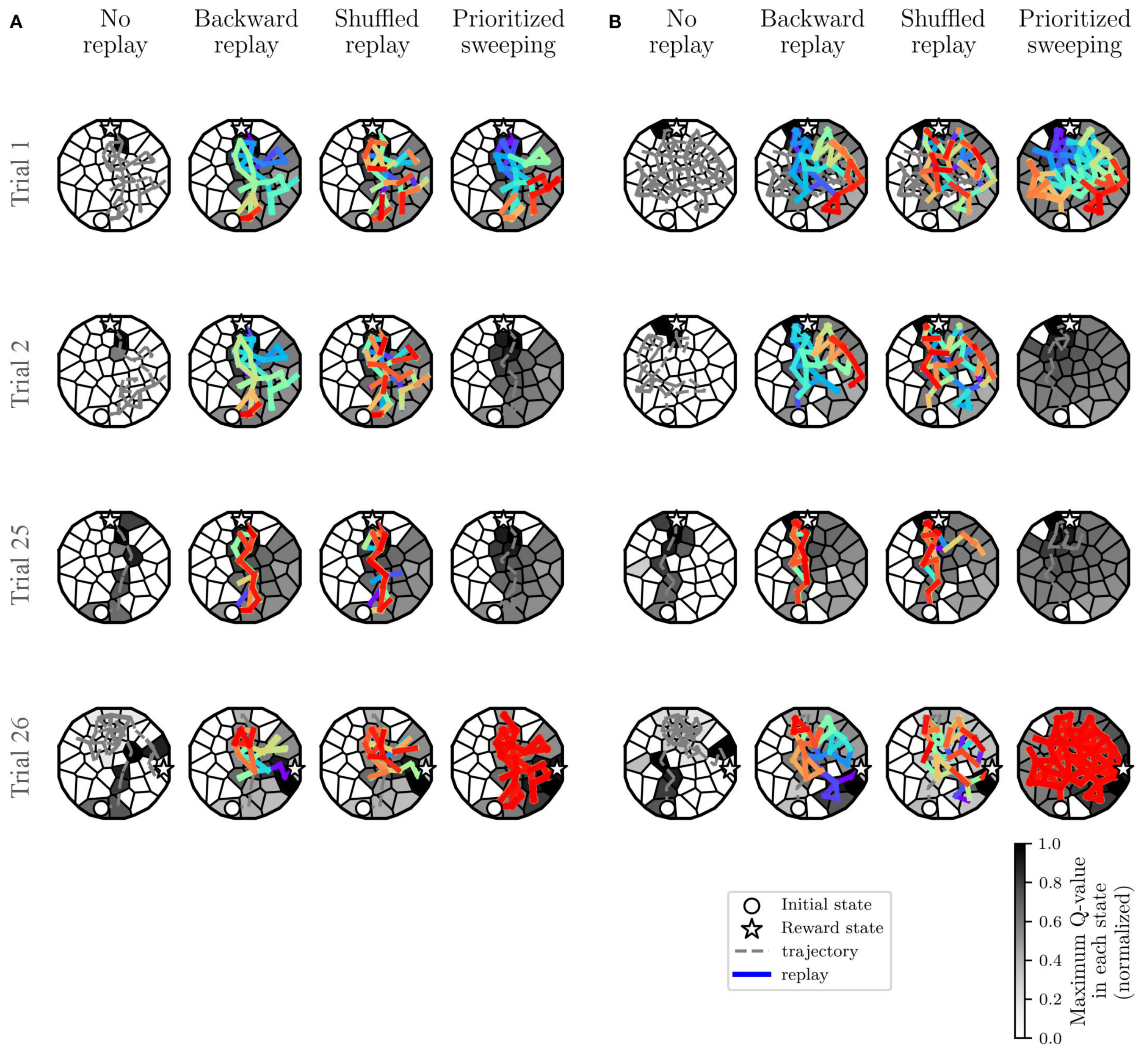


FIGURE 6 | Learning dynamics of the most representative individual: covered trajectory and replay at some critical trials. Also, for each state s, the $maxQ(s, a_i)$, among all the a_i, with i from 1 to 8 (**Figure 8A**, top right), is represented. The initial state and the reward state are also represented in the figure. **(A)** Experiments in the deterministic MDP. **(B)** Experiments in the stochastic MDP.

In both environments, as expected from the previously analyzed learning performance in **Figure 5**, there is no effective difference in terms of Q-value propagation between *MF-RL backward replay* and *MF-RL shuffled replay*. The explored trajectories and the replay are also very similar, resulting in not significantly different performances (**Figure 5**). These results, which simulate a spatial learning experiment for rodents (Morris, 1981) in a robotic framework, suggest some first advantageous properties of using replay-inspired strategies in neurorobotics. Our results imply that MF-RL replays could be sufficient to speed up learning and adaptation to non-stationarity (**Figure 5**, Trials 1–5 and 26–30), but MB-RL replay strategies could improve the adaptability of the system even more, with a higher level of stochasticity which often characterizes real robotic scenarios (**Figure 5**, Trials 26–30).

The proposed models and experiments contribute to a deeper understanding of the advantages and limitations of the existing RL models of replay in such robotics tasks. This experimental comparison, examining either a deterministic or stochastic version of the same environment (which implies a significantly different level of explored trajectories in the maze, see **Figure 3**) was useful to observe that RL replay gives an important contribution to a robotic spatial learning task, even if the model of the robot-environment interaction is stochastic. Nevertheless, a good compromise between the exploration capability of MF replay strategies and the adaptability of MB ones has not yet been found within these experiments. The next section will illustrate the performances of RL replay strategies in spatial learning when they are tested in combination in an MF-MB RL hybrid learning architecture in a more complex environment with obstacles, higher stochasticity, and non-stationarity.

4. COMBINING MB AND MF REPLAY IN A CHANGING ENVIRONMENT

Hippocampal replay has not only been interpreted as a memory consolidation process from past experience (Foster and Wilson, 2006; Girardeau et al., 2009), putatively MF, but also as a possible MB planning process that enables the mental simulation of hypothetical actions (Gupta et al., 2010; Ólafsdóttir et al., 2018; Khamassi and Girard, 2020). Along these lines, it has been argued that model sampling can not only be used for planning but also to update action values (van Seijen and Sutton, 2015; Cazé et al., 2018; Mattar and Daw, 2018). Moreover, some sequences of reactivated hippocampal neurons cannot be accounted for as a simple MF reactivation of past experience, and rather seem to represent creative combinations of past and experienced trajectories which can only be accounted for by a MB process (Gupta et al., 2010).

This suggests that both MF *MemR* and MB *SimR* are required to account for the diversity of hippocampal replays. Importantly, state-of-the-art models of RL processes in the mammalian brain assume a co-existence of MB and MF processes (Daw et al., 2005; Dollé et al., 2010, 2018; Keramati et al., 2011; Khamassi and Humphries, 2012; Pezzulo et al., 2013; Collins and Cockburn, 2020). Hence, neurorobotics constitutes a promising research area to study replay in robot control architectures that combine MB and MF RL processes.

With the experiments presented in the two previous sections, the complementary properties and performances of MF replay and MB replay have been analyzed. In our presented tasks, RL agents with MB replays tended to be slower to converge to an optimal solution but eventually, they reached a faster path to the reward location. On the other hand, the same agent with MF replay learned faster but converged to a suboptimal solution. In this section, in addition to pushing robot simulations toward more complex environments with stochasticity and non-stationarity, we want to examine the benefits of combining SimR and MemR in a robot control architecture which includes both MB and MF RL[3]. We thus investigate the effects of including replay in the algorithm proposed in Dromnelle et al. (2020b), which coordinates a MB and a MF RL expert within the decision layer of a robot control architecture. Interestingly, this algorithm had been previously tested in a navigation environment that includes open areas, corridors, dead-ends, a non-stationary task with changes in reward location, and a stochastic transition function between states of the task. In these conditions, previous results showed that the combination of MB and MF RL enables to (1) adapt faster to task changes thanks to the MB expert and (2) avoid the high computational cost of planning when the MF expert has been sufficiently trained by observation of MB decisions (Dromnelle et al., 2020b). Nevertheless, replay processes have not been included in this architecture yet, and the present paper is the opportunity to do it.

The results that we are going to illustrate and discuss in the following subsections present the combination of SimR and MemR as a critical resource to optimize the trade-off between the increase in performance and the reduction of computational cost in a hybrid MB-MF RL architecture when solving a more complex non-stationary navigation task than the two previous sections.

[3]The code for these simulations is available at https://github.com/elimas9/combining_MB_MF_replay.

4.1. Materials and Methods

The robot control architecture proposed in Dromnelle et al. (2020b) and also successfully applied to a simulated human-robot interaction task in Dromnelle et al. (2020a) takes inspiration from the mammalian brain's ability to coordinate multiple neural learning systems. Such ability is indeed considered to be key to making animals able to show flexible behavior in a variety of situations, to adapt to changes in the environment, while at the same time minimizing computational cost and physical energy (Renaudo et al., 2014). The proposed architecture in **Figure 7** is composed of a decision layer where a MF expert and a MB expert compete to determine the next action of the system. Both experts pass through three different phases: learning, inference, and decision. A meta-controller (MC) determines which proposed decision will be executed, following an arbitration criterion that we describe below.

4.1.1. Model-Based Expert

The model-based algorithm is implemented to learn a transition model T and a reward model R of the specific task. Thanks to these two learned models, it can predict the consequences of a given action several steps ahead and can adapt faster to non-stationary environments. Yet these computations are very costly (i.e., 1,000 times higher than the computations of the MF expert in Dromnelle et al., 2020b).

During the *learning process*, the transition model and the reward model are updated at each timestep after observing the departure state s of the robot, the action a that it has performed, the arrival state s', and the scalar reward r that this transition may have yielded. The transition model is updated by estimating $T(s, a, s')$, the probability of arriving in s' from (s, a), considering the past T_{tw} actions (**Table 3**). This probability is computed as already shown in Equation 3. Besides, the reward model $R(s, a, s')$ is updated by considering the most recent reward r_t associated to the transition (s, a, s'), multiplied by the probability of the transition itself in Equation 3.

The *inference process* estimates the action-value function *via* the value iteration algorithm (Sutton and Barto, 1998), and it operates as an offline planning phase that is continuously called every decision step, just before a decision is made by the agent about which action to perform. The maximal duration of this planning process can be determined either by setting a finite

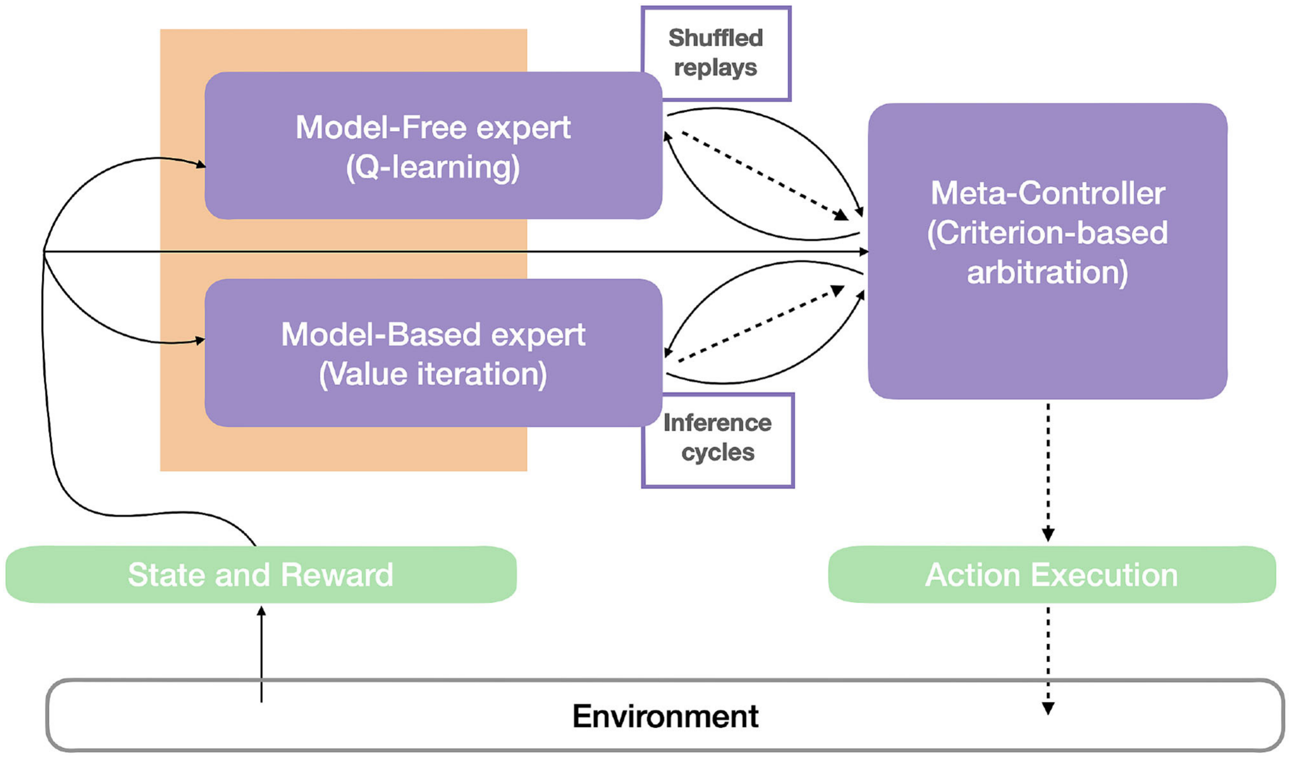

Shuffled replays
Model-Free expert (Q-learning)
Meta-Controller (Criterion-based arbitration)
Model-Based expert (Value iteration)
Inference cycles
State and Reward
Action Execution
Environment

FIGURE 7 | Robot control architecture. The agent-environment interaction can be described by (1) the state and the reward, as perceptual information (continuous arrows) from the environment, and (2) by the action (dashed lines) that the agent operates in the environment. The perceptual information is used by the model-free, the model-based expert, and the meta-controller (in purple). Based on this information and memory of their previous performances, the meta-controller estimates the entropy and computational cost of the experts, consistently with the criterion in Equation 13, and thus choose the expert that will be allowed to infer the probability distribution of the next agent's actions. This distribution, and the times consumed to compute it (dashed arrows), are then sent to the meta-controller. Different from Dromnelle et al. (2020b), both experts here have a 'replay' (reactivation) budget (limited or until convergence) that will affect both their performance and computation time and thus impact the meta-controller's arbitration. Here, shuffled memory reactivations (MemR) are integrated with the Q-learning algorithm of the MF expert, while simulation reactivations (SimR) constitute the offline MB inference iterations in the value iteration algorithm of the MB expert.

budget for the number of transitions over which the agent will evaluate its decision or by employing a convergence criterion based on the sum of the absolute action-value function estimation errors. More precisely, the planning terminates at iteration c if

$$\sum_{s,a} \left|\delta^c_{s,a}\right| < \epsilon_{MB} \quad \text{where} \tag{7}$$

$$\delta^c_{s,a} = \sum_{s'} p(s'|s,a)[R^c_{s,a} + \gamma V(s')^c] - Q(s,a)^c \tag{8}$$

Here, $R^c_{s,a}$ is the reward function of performing action a from state s at the offline reactivation c and $V(s')$ is the value function of the arriving state s' at reactivation c, from state s and action a. γ is the discount factor (**Table 3**).

Finally, the *decision process* chooses the next action to be performed by the robot by converting the action-value function into a probability distribution using a softmax function (see Equation 2), with an exploration/exploitation trade-off parameter β given in **Table 3**.

4.1.2. Model-Free Expert

The model-free algorithm does not learn any transition or reward model of the task, in contrast to the MB expert. Rather, it locally updates the current action-value function $Q(s,a)$ at each timestep. This property of the MF expert makes it save computational cost, compared to the MB expert, at the expense of slow adaptability to task changes, given the expert's lack of topological knowledge of the environment.

The *inference process* simply consists of reading from the Q-table the line corresponding to s which is then used by the *decision process*. The latter chooses the next action from the Q-values, also converted to a probability distribution with a β trade-off parameter in **Table 3**.

For the MF-RL expert, the *learning process* is defined as a tabular Q-learning algorithm in which the action-value function $Q(s,a)$ is updated according to Equation 1. Following the online learning phase, *shuffled replay* is performed, using the (s,a,s',r) tuples experienced by the agent in a given time-window of past transitions R_{tw} (**Table 3**). As for the MB expert, these offline updates stop when either the maximal predefined budget is exhausted or when the Q-values have converged. Since the MF

TABLE 3 | Parameters used to generate the results in this section.

	Model-based	Model-free	Meta-controller
α	-	0.6	-
γ	0.95	0.9	-
β	50	50	50
ϵ	0.01	0.1	-
RB	-	100	-
R_{tw}	-	100	-
T_{tw}	30	-	-

They are taken from Dromnelle et al. (2020b) as a starting point for this work. α is the learning rate, γ is the discount factor, and β is the exploration/exploitation trade-off parameter. For the MF expert, the converge threshold ϵ and replay constant RB have been introduced to design the convergence criterion, while ϵ for the MB expert is the same as in Dromnelle et al. (2020b). R_{tw} is the number of the last (s, a, s', r) tuples that the MF expert can replay. T_{tw} is the number of the last (s, a, s', r) tuples considered to built the transition model T for the MB expert.

expert does not know the transition probabilities of the task, a convergence test is computed for every offline learning iteration c as in Equation 9, where $act^c_{s,a} = \tau \cdot act^{c-1}_{s,a}$, with $act^{\tilde{c}_{s,a}}_{s,a} = RB$ during the first time $\tilde{c}_{s,a}$ when that specific transition is selected for replay and with $act^0_{s,a} = 0$. act is an activation function defined for each couple (s, a), and it is 0 if (s, a) has not been replayed before or otherwise it decays from RB (**Table 3**) along the replay iterations c with a time constant τ (Equation 11).

$$\sum_{s,a} \delta^c_{s,a} act^c_{s,a} < \epsilon_{MF} \quad \text{where} \tag{9}$$

$$\delta^c_{s,a} = |Q(s,a)^c - Q(s,a)^{c-1}| \tag{10}$$

The principle behind the design of this convergence criterion is that the importance of each $\delta_{s,a}$ (Equation 10) starts as RB and decreases over the offline learning iterations c, following the decay constant τ (Equation 11). This strategy does not constrain the number of needed replay iterations, because the agent would still perform replays due to high $\sum_{s,a} \delta^c_{s,a} act^c_{s,a}$. Nevertheless, this value will slowly decrease the need for more replay iterations along with the offline learning phase. RB is a value representing one of the possible replay budgets needed to obtain performances that are comparable to the maximum amount of reward that the expert can collect, thus not inhibiting the offline learning phase when needed. Finally, the convergence threshold ϵ_{MF} is an order of magnitude larger than ϵ_{MB} (**Table 3** which is the same used in Dromnelle et al., 2020b). The MF expert does not know the probabilities contained in the transitions model in Equation 3 and for this reason, its convergence criterion is based on the actual update of the action-value function $Q(s, a)$. This means that, in the MF case, the $\delta^c_{s,a}$ are not multiplied by any probability, derived from the world model, and thus their values will usually be an order of magnitude larger than the $\delta^c_{s,a}$ of the MB case, multiplied instead by the probability of a given (s, a, s', r) tuple.

$$\tau = \sqrt[RB]{\frac{\epsilon_{MF}}{RB}} \tag{11}$$

4.1.3. Meta-Controller

The meta-controller selects, which expert will take the control of the next action, by following a specific criterion that is a trade-off between the learning performances and the computational cost of the inference process of the two agents and it is called *Entropy and Cost (EC)* (Dromnelle et al., 2020b).

On the one hand, the quality of learning is computed by Equation 12 where $f(P(a|s, E, t)$ is a low-pass filtered action probability distribution with a time constant $\tau = 0.67$, previously used as an indicator of the learning quality in humans (Viejo et al., 2015).

$$H_{exp}(s,E,t) = -\sum_{a=0}^{|\mathcal{A}|} f(P(a|s,E,t)) \cdot \log_2 (f(P(a|s,E,t))) \tag{12}$$

On the other hand, the cost of the process $C(s, E, t)$ is the computation time needed to perform the inference phase for the expert E, at time t, and it is also filtered as the action probability distribution above.

Eventually, the MC chooses which expert will take control of the next decision by following the equation below (Dromnelle et al., 2020b):

$$EX(s,E,t) = -(H_{exp}(s,E,t) + \kappa C(s,E,t)) \tag{13}$$

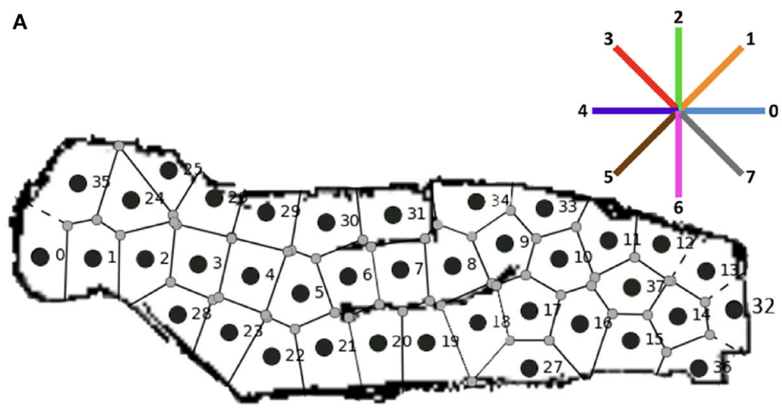
$EX(s, E, t)$ is the expertise value of the expert E, which is then converted into a distribution of probabilities using a softmax function. κ weights the impact of time in the criterion by assigning greater importance to the computation time when the entropy component $H_{exp}(s, E, t)$ of the MF experts is low.

After applying Equation 13, the MC draws the winning expert from the softmax of the distribution of their expertise $EX(s, E, t)$ (with a trade-off coefficient β shown in **Table 3**) and inhibits the inference process of the expert that is not selected.

4.1.4. The Experimental Set-Up and Implementation

This new hybrid MB-MF RL architecture with replay is tested in a dynamic navigation task where the robot has to learn how to reach a unitary rewarding state. The task remains stationary during the first 1,600 over 4,000 iterations and then the reward is moved to another state (from state 18 to state 34, **Figure 8**). In this experiment, an extra element of non-stationarity is represented by the starting state of the robot being uniformly selected with the same probability between state 0 and state 32 at the beginning of each trial (**Figure 10**). Different from Dromnelle et al. (2020b), experiments where the reward is fixed or where a new obstacle is introduced have not been performed for this work.

First, the real Turtlebot autonomously navigates within the environment using a SLAM Gmapping algorithm (**Figure 8**) and creates a discrete map of the maze (38 Markovian states are identified and shown in **Figure 8A**). This autonomous state decomposition process is identical to the one used in the previous experiment described in Section 3.1.2. The robot-environment ratio is very similar to the one of the previous experiment in



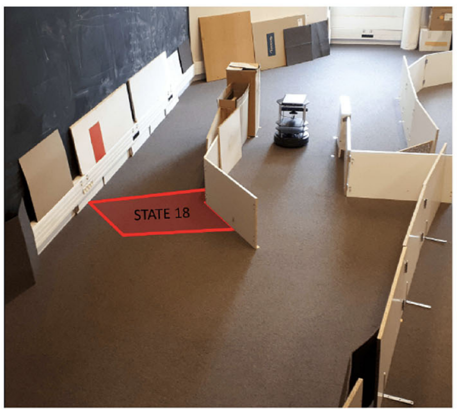


FIGURE 8 | Description of the experimental set-up. **(A)** Map of the discrete states of the maze. The eight-pointed star indicates the cardinal directions in which the robot can move. These directions are the same used for the experiment in Section 3 **(B)** Photo of the real Turtlebot approaching the initial rewarding state 18, highlighted in the figure. Adapted from Dromnelle et al. (2020b).

Section 3.1.2: the state radius is 35 cm, in this case, and the robot size is 35,4 x 35,4 x 42 (L x W x H, cm).

Then, during a second free exploration phase, the robot learns the transition model of the environment, that is, the probability that the robot starts its move in one state s performs an action a and arrives in another state s'. This second phase of the creation of the transition model is also conducted as in Section 3, but with the real robot in this case.

After these exploration phases, the subsequent experiments involving a reward were performed in simulation to test the impact of different parameters of the algorithm and study the effect of replay on total performance and computation cost. During these simulations, the agent experienced the MDP based on the transition map that was empirically acquired with the real robot (as was done in Dromnelle et al., 2020b).

Figure 10B shows the maximum level of uncertainty for each of the 38 states of the environment. This uncertainty is computed in the same way as for the other experiment in Equation 5, and the transitions map is used to guide the robotic exploration in the simulation environment.

The action space is also discrete and consists of 8 possible cardinal directions equally distributed around the agent. Given the discrete and probabilistic nature of the state and action spaces, the transition model $T(s, a, s')$ (Equation 3) and the reward model $R(s, a)$ of the MB expert are probability distributions.

4.2. Results

To evaluate the contribution of combining MB and MF replay in terms of performance and computational cost, we tested several algorithms. First, we are interested in simulating the two baseline cases, pure MF and pure MB algorithms, and how they perform with the respective MemR and SimR and limited budgets. Finally, we want to test the combination of the two strategies by using the criterion proposed in Dromnelle et al. (2020b), with either an infinite or a limited reactivations budget. Here are the relevant combinations of the same controller that we tested in this task:

- MF only agent, no replay
- MF only agent with MF replay (infinite replay budget)
- MF only agent with MF replay (budget: 200 replay iterations)
- MB only agent with MB replay (infinite inference budget)
- MB only agent with MB replay (budget: 200 inference iterations)
- MB+MF agent with MB replay (infinite inference budget)
- MB+MF agent with MB budget (budget: 200 inference iterations)
- MB+MF agent with MF replay (budget: 100 replay iterations) and MB replay (budget: 100 inference iterations) (a fair comparison with the previous cases because here the reactivation buffer is split in a maximum of 100 iterations per expert)

All the MB+MF agents use the EC coordination criterion described in Section 4.1.3. This criterion was taken from Dromnelle et al. (2020b) which showed that it allows for advantageous coordination between MB and MF experts and significantly reduces the computational cost of the inference phase, without relevantly impacting the amount of gained reward. **Table 3** shows the values of the parameters that we used for these experiments.

The speed of learning of all the above-listed agents was impacted when the reward's position changed at iteration #1600 (**Figure 9A**). It is interesting to notice that the *MB - inference budget 100 + MF - replay budget 100* agent, which exploited the EC criterion with a limited budget for the two experts, shows a faster increase in the cumulative reward compared to all the other agents, from around actions #2500. As observed in the previous experiment (Section 3), replay contributes to increasing the speed of learning and by combining the action of both MF and MB replay, it is possible to better account for both adaptability and generalization, drastically leading to a steeper accumulated reward over time slope of the proposed strategy, without having the same growth on the computational cost side (**Figures 9A,B**). Concerning the cumulative costs, **Figure 9B**

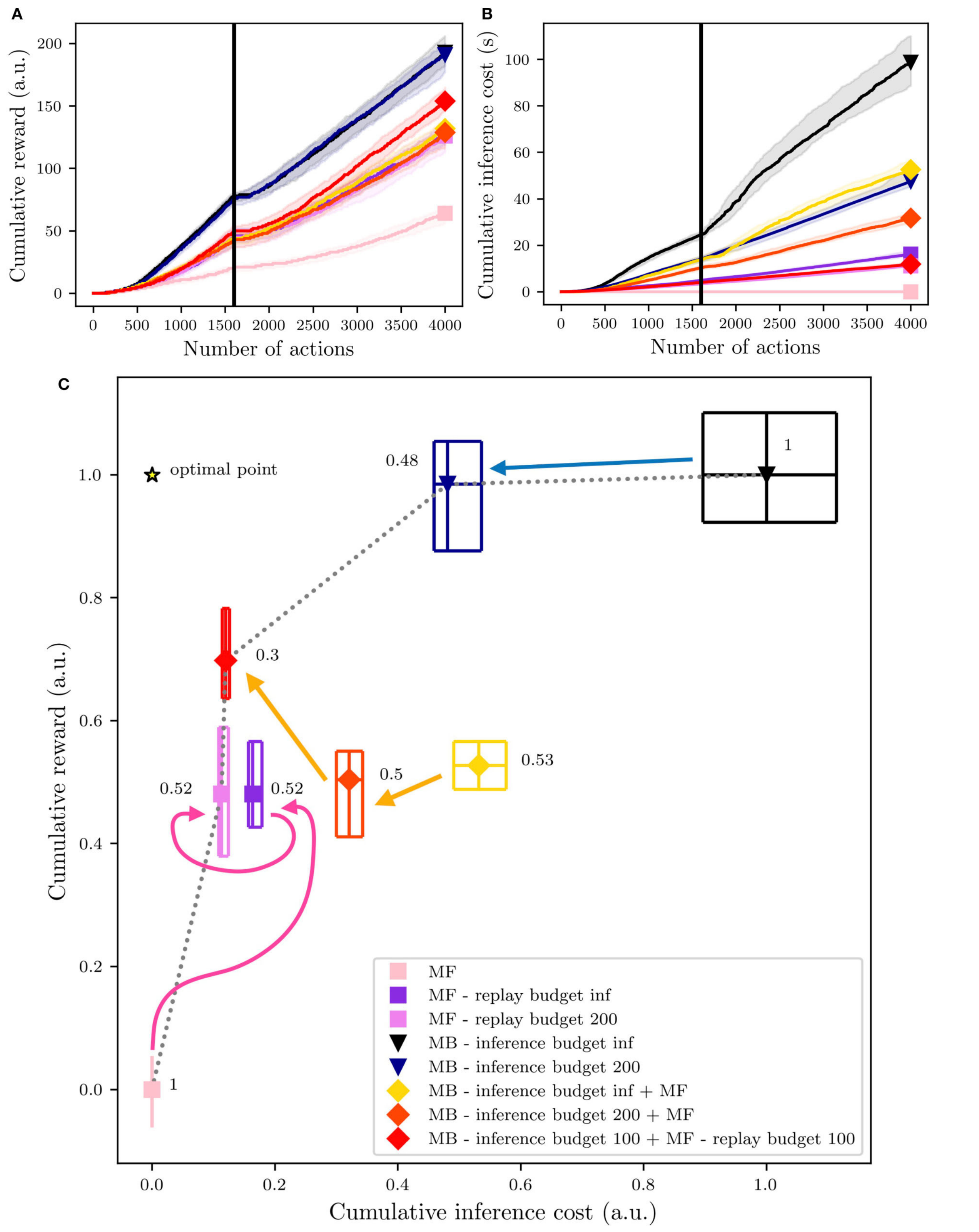


FIGURE 9 | Overall performances of the different agents during their first 4,000 actions in the environment. The vertical black line highlights the trial when the reward switch (1,600). **(A)** The dynamics of the reward's accumulation. **(B)** The dynamics of the computational cost's accumulation. **(C)** An overview of the algorithms' position within a normalized reward × cost space. The central polygons represent the median of the performance over 50 simulated experiments. Cumulative rewards and costs have been normalized considering that the MF medians of the cumulative rewards and costs correspond to 0 and that the MB medians of cumulative rewards and cost correspond to 1.

shows that it rapidly increases for the *MB - inference budget inf* agent when the environment changed, and eventually, by action #4000, its cumulative cost has doubled the ones of the other agents.

Thus, considering the final overview of the performances and computational costs in **Figure 9C**, deeper analyses and comparisons of the tested algorithms can be presented. The results are represented in terms of first, median, and third percentiles over 50 experiments. The cumulative reward is the amount of reward each agent has accumulated over the entire experiment, which is composed of 4,000 iterations of the learning, inference, and decision processes together (Section 4.1). The cumulative inference cost represents the time (in seconds) needed to perform the inference phase.

As expected, reward-wise, the best performing agent is the pure MB, with an infinite inference budget (black triangle, on the top-right, in **Figure 9C**). However, this agent is also the most costly in terms of computation during the inference phase. This issue can be partially fixed by reducing the MB replay budget to 200 iterations (blue triangle, in **Figure 9C**). In this case, the inference phase will be stopped either if the action values have converged or if the number of inference iterations has reached the maximal budget (in this case 200).

On the opposite side of the figure, the pure MF agent (pink square, on the bottom-left, in **Figure 9C**) shows the minimum cost of the entire set of experiments, but also the lowest cumulative reward. Adding replay to the MF expert, with an infinite replay budget (dark violet square in **Figure 9C**) or a 200-iteration budget (light violet square in **Figure 9C**) doubles the reward accumulation performance, with a limited increase in the computational costs (compared to the MB costs), in particular when adding the budget of 200 iterations.

From the results in **Figure 9C**, we can deduce that for both the MF and the MB experts, most of the time, the number of needed reactivations is in the same order of magnitude as the proposed finite budget of 200 (since the cumulative costs are comparable). As already shown in Dromnelle et al. (2020b), with an MB expert with an infinite inference budget, the coordination of MB and MF experts *via* the EC criterion produces agents which are halfway between MB-only and MF-only experts, regarding performances and costs (yellow diamond in **Figure 9C**). Nevertheless, when limiting the MB inference budget to 100 and adding the contribution of 100 replay iterations for the MF expert (red diamond in **Figure 9**), the cumulative reward increases, and the inference cost diminishes, moving the performance of the agent closer to the optimal point (star in **Figure 9**). Moreover, the arrows highlight the progressions of the *MF-only* (pink), the *MB-only* (blue), and the *MB+MF* (orange) agents. Looking in more detail, the performance of the MF-only agents is improved by adding a budget of 200 MF replays and on the other hand, the performance of the MB-only agents is slightly decreased by limiting the inference budget to 200 iterations, but the cumulative computational cost is significantly decreased. Starting from the performance obtained in Dromnelle et al. (2020b), in yellow in the figure, we obtain similar performances but decrease the computational cost when we limited the inference budget to 200 inference iterations for the MB expert, producing agents which are halfway between MB-only and MF-only experts. After this analysis, we have tested the combination of the two best strategies tried so far: the MB expert with a limited inference budget and the MF one with a limited replay budget and we have combined them through the EC criterion (Equation 13). In this case, to have the same total reactivations budget as the other tested algorithm, we have shared the initial 200 reactivations budget to 100 SimR for the MB expert and 100 MemR for the MF one. With this combined replay effort, the overall performance reached an optimal compromise between performance and cost since the inference cost is substantially decreased while the cumulative reward was significantly raised, compared to the results obtained by Dromnelle et al. (2020b).

Given that the aim of each agent and its EC meta-controller is composed of two objectives: (1) maximizing the cumulative reward and (2) minimizing the cumulative inference cost, we compute the pareto front (black dotted line in **Figure 9C**), which represents the solutions that approximate the set of all optimal trade-offs of the two given objectives. As expected, the pure MB and MF experts are pareto optimal solutions, very specialized in one of the two objectives, while by reducing and splitting their budgets we can have agents that interestingly converge closer to the *OptimalPoint* (star in **Figure 9C**). To rank all the agents *ag*, the Chebyshev distance (Cantrell, 2000) from their median performance to the *OptimalPoint* is computed as shown in the following equation:

$$\textit{Chebyschev distance}\,(ag) = \max_{obj} \mid \textit{OptimalPoint}_{obj} - \textit{median}(ag_{obj}) \mid \tag{14}$$

where *obj* are the 2 normalized objectives of the solutions space (cumulative inference costs and cumulative reward). The computed Chebyshev distances are shown in **Figure 9C**, on the side of each algorithm point, and show a clear picture concerning the proposed solutions; the agent sharing the reactivations budget between the MB and MF is the closest to the optimal point, followed by the MB expert with limited SimR budget. MF with MemR and MB + MF without MemR have very similar distances to the optimal points, meaning that the contribution of the MB expert is key to adapting to a dynamical environment, but the cost of this computation can largely decrease just when it cooperates with an MF agent with replay, that can learn faster also from the Q-values update of the MB expert.

These results open new possibilities for the design of RL control architectures in robotics. When dealing with probabilistic environments, MF replay might focus mainly on rare and not relevant transitions, leading to interesting exploration and computational economy, but misguiding the memory consolidation of relevant experience, when changes happen in the task (as also seen in Section 3). When the transitions model is stochastic, the combination of the computationally competitive MF replay with the general knowledge of the environment, acquired by MB replay, can bring artificial agents and robots to better deal with a non-stationary RL task.

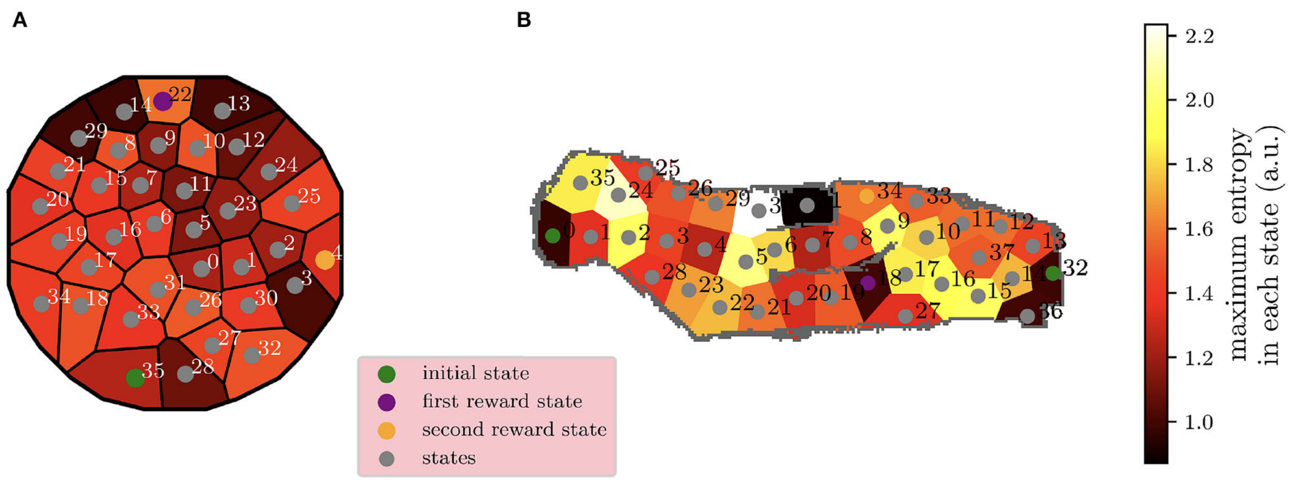


FIGURE 10 | Representation of the navigation environments for the previous experiments, Section 3 and Section 4, organized in, respectively, 36 and 38 discrete Markovian states decomposed from the data acquired during the autonomous navigation of the robot, when no reward was present in the mazes. The initial and reward states for the tasks are also highlighted in the figure. In these heatmaps, the lighter the color of the state, the greater the maximal entropy of that specific state, according to Equation 5. The represented scale of entropy values (0.87–2.23 a.u.) has been selected to cover the whole range of the computed entropies. Moreover, in both environments, the robots have navigated for 5357 actions. **(A)** In the case of the circular maze (Section 3), the navigation and the transition model are acquired after simulated navigation on ROS Gazebo. **(B)** In the second experiment (Section 4), the navigation and the transition model are instead computed after the real robot navigation, which generated a wider range of maximal entropy values, sometimes also very low due to the presence of walls that categorically constrained certain states of the environment.

5. DISCUSSION

In this paper, our research question was whether RL strategies using neuro-inspired replay methods, based on neuroscience knowledge about the hippocampal replay, could improve the speed and the adaptability of robotic agents engaged in spatial navigation tasks. MF, MB, and no replay RL techniques were compared in three simulated robotic experiments of increasing complexity and realism. Our results showed that in all levels of abstraction, the neurorobots learned the spatial task faster when the replay was involved in the process, and more efficiently when a MB method replay method was used. Conversely, we show how a synergy between MB and MF replay methods can be more effective in a more realistic and stochastic experimental setup.

The application of RL techniques to robotics requires coping with some specificities of operating in the real world (Kober et al., 2013). First, making actual movements in the real world takes time, wares out the robotic platform, and also has the potential of damaging it. Acquiring new data requires the robot to move, and thus to incur those costs. Online learning processes therefore have to be as much parsimonious on data use as possible. Second, making decisions also takes time, especially when using limited embedded computation systems, while operating in a dynamic world may require the ability to react extremely rapidly to avoid damage. Learning systems should thus be as computationally cheap as possible. Finally, moving and computing both consume the robot's energy, which is always available in limited amounts. This highlights the importance of developing robotic controllers that can (1) maximize their learning capabilities over experience and energy scarcity and (2) reduce the complexity of their algorithm to meet the computational limitations of embedded platforms.

All along with this paper, we have presented simulated experiments (sometimes based on data like transition maps first generated with a real robot) to investigate the possible advantages of equipping neurorobots with offline learning mechanisms inspired by hippocampal place cells' reactivations. These advantages are, first, to extract as much information as possible from the already gathered data, and, by mixing the multiple types of learning processes with the multiple types of reactivations, to limit deliberation time, and to limit the aforementioned costs intrinsic to robotics. Starting with simpler and deterministic environments, as the double T-maze experiment presented in Section 2, this research illustrates that as the complexity of the state-action spaces increases, MB *SimR* become more strategic for the learning capabilities of the agent (Section 3). In Section 4, the combination of MF *MemR* and MB *SimR* is presented as an interesting proposal to merge the benefits of both techniques: prioritizing the MB expert when the task requires more inference and generalization effectiveness to be solved (for example facing non-stationarity), while on the contrary giving priority to the MF expert when an effective solution can be found relying only on recent experience.

When simulations increase in complexity, thus getting closer to a real robotic experiment, the challenges regarding the internal representation of the world (in particular the state-action space and the reward) increase. As presented in **Figure 10**, where the environments of the two last experiments (presented in Sections 3, 4 respectively) are displayed in terms of maximum entropy per state, it is visible that the transition probability matrix created by the navigation of the real robot (**Figure 10B**) results in a representation of the environment which is less homogeneous and more uncertain than the one learned with the simulated robot (**Figure 10A**). Often, in mobile robotics, localization may depend on a few sensory information, as in the case of the mobile robots used in our experiments. Such limited information is however fundamental for the acquisition of a solid representation of the environment. For these reasons, the entropy maps in **Figure 10** reflect the nature of the two mazes: the uncertainty is more homogeneous in the circular maze (**Figure 10A**) since the environment is an open space which gives the agent an even chance to end up visiting the neighboring states. In contrast, the second environment (**Figure 10B**) is longer in one dimension and presents inner walls that result in a fuzzier level of uncertainty on the transitions model of the environment.

Future works in this research direction would include the comparison with the RL algorithms performing forward replays, which are of crucial importance in standard rodents navigation tasks, such as the multiple T-maze (Johnson and Redish, 2007). These forward-shifted spatial representations have been demonstrated to happen largely at decision points to predict the consequences of the next actions. Their effect has already been successfully modeled in neurorobotics by Maffei et al. (2015), where they implemented the extractions of relevant policies by consulting memory. On the other hand, van Seijen and Sutton (2015) argued that it is mathematically equivalent to update Q-values in a MF way combined with replay and to update Q-values in a MB way, given that the elements in the memory buffer, used for replay, are the same than those used to build the model. Moreover, RL-based replay strategies can also generate forward replay events (Khamassi and Girard, 2020) and enable RL-based models to still account for neurobiological data (Cazé et al., 2018; Mattar and Daw, 2018).

In summary, this work presented new and crucial results concerning the advantages and the limitations of different RL-based replay techniques for robotics, gradually testing them in more and more complex and realistic circumstances. Additionally, this research paves the way for new studies on the role of replays in neurorobotics, in particular, in spatial navigation tasks where generalization effectiveness and time efficiency are key.

Finally, the addition of RL techniques, inspired by hippocampal replays, shows an improvement in the performance of the presented navigation task, in particular, concerning the exploitation of the past experience, knowledge propagation, and as a consequence, the speed of learning. MB *SimR* significantly contributed in the case of non-stationarity, but a fruitful coordination with MF *MemR* became crucial in terms of computational cost reduction. All these insights, found in robotic experiments, implemented with different levels of abstraction, can encourage new neuroscientific experimental protocols and shed light on a better understanding of the phenomenon of hippocampal replay.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

BG and MK: conceptualization, project administration, and funding acquisition. EM, JB, JM, RD, BG, and MK: methodology. EM, JB, JM, RD, JC, EP, and MK: software. EM and MK: validation and writing—original draft. EM and JB: formal analysis. EM, JB, JM, BG, and MK: writing—review and editing. EM, JB, JM, EP, and MK: visualization. EM, BG, and MK: supervision. All authors contributed to the article and approved the submitted version.

FUNDING

The authors acknowledge the funding of the CNRS 80|PRIME program, RHiPAR project, the French National Research Agency ANR, and the Délégation Générale de l'Armement (RD).

ACKNOWLEDGMENTS

We thank Michaël Zugaro and Natalia Díaz Rodríguez for the constructive dialogue that we had during the key steps of this work.

REFERENCES

Arleo, A., and Gerstner, W. (2000). Spatial cognition and neuro-mimetic navigation: a model of hippocampal place cell activity. *Biol. Cybern*. 83, 287–299. doi: 10.1007/s004220000171

Aubin, L., Khamassi, M., and Girard, B. (2018). "Prioritized sweeping neural DynaQ with multiple predecessors, and hippocampal replays," in *Conference on Biomimetic and Biohybrid Systems* (Paris: Springer), 16–27.

Benchenane, K., Peyrache, A., Khamassi, M., Tierney, P. L., Gioanni, Y., Battaglia, F. P., et al. (2010). Coherent theta oscillations and reorganization of spike timing in the hippocampal-prefrontal network upon learning. *Neuron* 66, 921–936. doi: 10.1016/j.neuron.2010.05.013

Caluwaerts, K., Staffa, M., N'Guyen, S., Grand, C., Dollé, L., Favre-Félix, A., et al. (2012). A biologically inspired meta-control navigation system for the psikharpax rat robot. *Bioinspiration Biomimet*. 7, 025009. doi: 10.1088/1748-3182/7/2/025009

Cantrell, C. D. (2000). *Modern Mathematical Methods for Physicists and Engineers*. Cambridge: Cambridge University Press.

Cazé, R., Khamassi, M., Aubin, L., and Girard, B. (2018). Hippocampal replays under the scrutiny of reinforcement learning models. *J. Neurophysiol*. 120, 2877–2896. doi: 10.1152/jn.00145.2018

Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A., and Fiete, I. (2019). The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nat. Neurosci*. 22, 1512–1520. doi: 10.1038/s41593-019-0460-x

Collins, A. G., and Cockburn, J. (2020). Beyond dichotomies in reinforcement learning. *Nat. Rev. Neurosci*. 21, 576–586. doi: 10.1038/s41583-020-0355-6

Daw, N. D., Niv, Y., and Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nat. Neurosci*. 8, 1704–1711. doi: 10.1038/nn1560

De Lavilléon, G., Lacroix, M. M., Rondi-Reig, L., and Benchenane, K. (2015). Explicit memory creation during sleep demonstrates a causal role of place cells in navigation. *Nat. Neurosci*. 18, 493–495. doi: 10.1038/nn.3970

Diba, K., and Buzsáki, G. (2007). Forward and reverse hippocampal place-cell sequences during ripples. *Nat. Neurosci*. 10, 1241–1242. doi: 10.1038/nn1961

Dollé, L., Chavarriaga, R., Guillot, A., and Khamassi, M. (2018). Interactions of spatial strategies producing generalization gradient and blocking: a computational approach. *PLoS Comput. Biol*. 14, e1006092. doi: 10.1371/journal.pcbi.1006092

Dollé, L., Khamassi, M., Girard, B., Guillot, A., and Chavarriaga, R. (2008). "Analyzing interactions between navigation strategies using a computational model of action selection," in *International Conference on Spatial Cognition* (Springer), 71–86.

Dollé, L., Sheynikhovich, D., Girard, B., Chavarriaga, R., and Guillot, A. (2010). Path planning versus cue responding: a bio-inspired model of switching between navigation strategies. *Biol. Cybern*. 103, 299–317. doi: 10.1007/s00422-010-0400-z

Dromnelle, R., Girard, B., Renaudo, E., Chatila, R., and Khamassi, M. (2020a). "Coping with the variability in humans reward during simulated human-robot interactions through the coordination of multiple learning strategies," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)* (Naples: IEEE), 612–617.

Dromnelle, R., Renaudo, E., Pourcel, G., Chatila, R., Girard, B., and Khamassi, M. (2020b). "How to reduce computation time while sparing performance during robot navigation? a neuro-inspired architecture for autonomous shifting between model-based and model-free learning," in *Conference on Biomimetic and Biohybrid Systems* (Freiburg: Springer), 68–79.

Ego-Stengel, V., and Wilson, M. A. (2010). Disruption of ripple-associated hippocampal activity during rest impairs spatial learning in the rat. *Hippocampus* 20, 1–10. doi: 10.1002/hipo.20707

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., et al. (2020). "Revisiting fundamentals of experience replay," in *International Conference on Machine Learning* (Vienna: PMLR), 3061–3071.

Fleischer, J. G., Gally, J. A., Edelman, G. M., and Krichmar, J. L. (2007). Retrospective and prospective responses arising in a modeled hippocampus during maze navigation by a brain-based device. *Proc. Natl. Acad. Sci. U.S.A*. 104, 3556–3561. doi: 10.1073/pnas.0611571104

Foster, D. J., and Wilson, M. A. (2006). Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature* 440, 680–683. doi: 10.1038/nature04587

Girardeau, G., Benchenane, K., Wiener, S. I., Buzsáki, G., and Zugaro, M. B. (2009). Selective suppression of hippocampal ripples impairs spatial memory. *Nat. Neurosci*. 12, 1222–1223. doi: 10.1038/nn.2384

Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. Rob*. 23, 34–46. doi: 10.1109/TRO.2006.889486

Gupta, A. S., van der Meer, M. A., Touretzky, D. S., and Redish, A. D. (2010). Hippocampal replay is not a simple function of experience. *Neuron* 65, 695–705. doi: 10.1016/j.neuron.2010.01.034

Jadhav, S. P., Kemere, C., German, P. W., and Frank, L. M. (2012). Awake hippocampal sharp-wave ripples support spatial memory. *Science* 336, 1454–1458. doi: 10.1126/science.1217230

Jauffret, A., Cuperlier, N., and Gaussier, P. (2015). From grid cells and visual place cells to multimodal place cell: a new robotic architecture. *Front. Neurorobot*. 9, 1. doi: 10.3389/fnbot.2015.00001

Ji, D., and Wilson, M. A. (2007). Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nat. Neurosci*. 10, 100–107. doi: 10.1038/nn1825

Johnson, A., and Redish, A. D. (2007). Neural ensembles in ca3 transiently encode paths forward of the animal at a decision point. *J. Neurosci*. 27, 12176–12189. doi: 10.1523/JNEUROSCI.3761-07.2007

Karlsson, M. P., and Frank, L. M. (2009). Awake replay of remote experiences in the hippocampus. *Nat. Neurosci*. 12, 913–918. doi: 10.1038/nn.2344

Keramati, M., Dezfouli, A., and Piray, P. (2011). Speed/accuracy trade-off between the habitual and the goal-directed processes. *PLoS Comput. Biol*. 7, e1002055. doi: 10.1371/journal.pcbi.1002055

Khamassi, M. (2007). *Complementary roles of the rat prefrontal cortex and striatum in reward-based learning and shifting navigation strategies* (Ph.D. thesis). Université Pierre et Marie Curie-Paris VI.

Khamassi, M., and Girard, B. (2020). Modeling awake hippocampal reactivations with model-based bidirectional search. *Biol. Cybern*. 114, 231–248. doi: 10.1007/s00422-020-00817-x

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

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

Kruskal, W. H., and Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *J. Am. Stat. Assoc*. 47, 583–621. doi: 10.1080/01621459.1952.10483441

Lesaint, F., Sigaud, O., Flagel, S. B., Robinson, T. E., and Khamassi, M. (2014). Modelling individual differences in the form of pavlovian conditioned approach responses: a dual learning systems approach with factored representations. *PLoS Comput. Biol*. 10, e1003466. doi: 10.1371/journal.pcbi.1003466

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn*. 8, 293–321. doi: 10.1007/BF00992699

Maffei, G., Santos-Pata, D., Marcos, E., Sánchez-Fibla, M., and Verschure, P. F. (2015). An embodied biologically constrained model of foraging: from classical and operant conditioning to adaptive real-world behavior in dac-x. *Neural Netw*. 72:88–108. doi: 10.1016/j.neunet.2015.10.004

Mattar, M. G., and Daw, N. D. (2018). Prioritized memory access explains planning and hippocampal replay. *Nat. Neurosci*. 21, 1609–1617. doi: 10.1038/s41593-018-0232-z

Michon, F., Sun, J.-J., Kim, C. Y., Ciliberti, D., and Kloosterman, F. (2019). Post-learning hippocampal replay selectively reinforces spatial memory for highly rewarded locations. *Curr. Biol*. 29, 1436–1444. doi: 10.1016/j.cub.2019.03.048

Milford, M., and Wyeth, G. (2010). Persistent navigation and mapping using a biologically inspired slam system. *Int. J. Rob. Res*. 29, 1131–1153. doi: 10.1177/0278364909340592

Moore, A. W., and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Mach. Learn*. 13, 103–130. doi: 10.1007/BF00993104

Morris, R. G. (1981). Spatial localization does not require the presence of local cues. *Learn. Motiv*. 12, 239–260. doi: 10.1016/0023-9690(81)90020-5

O'Keefe, J., and Dostrovsky, J. (1971). The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain Res*. 34, 171–175. doi: 10.1016/0006-8993(71)90358-1

Ólafsdóttir, H. F., Bush, D., and Barry, C. (2018). The role of hippocampal replay in memory and planning. *Curr. Biol*. 28, R37-R50. doi: 10.1016/j.cub.2017.10.073

Peng, J., and Williams, R. J. (1993). Efficient learning and planning within the dyna framework. *Adapt. Behav*. 1, 437–454. doi: 10.1177/105971239300100403

Pezzulo, G., Kemere, C., and Van Der Meer, M. A. (2017). Internally generated hippocampal sequences as a vantage point to probe future-oriented cognition. *Ann. N. Y. Acad. Sci*. 1396, 144–165. doi: 10.1111/nyas.13329

Pezzulo, G., Rigoli, F., and Chersi, F. (2013). The mixed instrumental controller: using value of information to combine habitual choice and mental simulation. *Front. Psychol*. 4, 92. doi: 10.3389/fpsyg.2013.00092

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., et al. (2009). "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software, Volume 3* (Kobe), 5.

Redish, A. D. (2016). Vicarious trial and error. *Nat. Rev. Neurosci*. 17, 147–159. doi: 10.1038/nrn.2015.30

Renaudo, E., Girard, B., Chatila, R., and Khamassi, M. (2014). "Design of a control architecture for habit learning in robots," in *Conference on Biomimetic and Biohybrid Systems* (Springer), 249–260.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*. doi: 10.48550/arXiv.1511.05952

Singer, A. C., and Frank, L. M. (2009). Rewarded outcomes enhance reactivation of experience in the hippocampus. *Neuron* 64, 910–921. doi: 10.1016/j.neuron.2009.11.016

Sutton, R. S. (1990). “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” in *Machine Learning Proceedings 1990* (Cambridge, MA: Elsevier), 216–224.

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

Tolman, E. C. (1939). Prediction of vicarious trial and error by means of the schematic sowbug. *Psychol. Rev*. 46, 318. doi: 10.1037/h0057054

Valenti, O., Mikus, N., and Klausberger, T. (2018). The cognitive nuances of surprising events: exposure to unexpected stimuli elicits firing variations in neurons of the dorsal ca1 hippocampus. *Brain Struct. Funct*. 223, 3183–3211. doi: 10.1007/s00429-018-1681-6

van Seijen, H., and Sutton, R. (2015). “A deeper look at planning as learning from replay,” in *International Conference on Machine Learning* (Lille: PMLR), 2314–2322.

Viejo, G., Khamassi, M., Brovelli, A., and Girard, B. (2015). Modeling choice and reaction time during arbitrary visuomotor learning through the coordination of adaptive working memory and reinforcement learning. *Front. Behav. Neurosci*. 9, 225. doi: 10.3389/fnbeh.2015.00225

Watkins, C. (1989). *Learning from delayed rewards* (Unpublished doctoral dissertation), Cambridge University, Cambridge.

Wilson, M. A., and McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science* 265, 676–679. doi: 10.1126/science.8036517

Zhang, S., and Sutton, R. S. (2017). A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*. doi: 10.48550/arXiv.1712.01275

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

frontiers | Frontiers in Neurorobotics

TYPE Methods
PUBLISHED 31 August 2022
DOI 10.3389/fnbot.2022.873239

Check for updates

OPEN ACCESS

EDITED BY
Florian Röhrbein,
Technische Universität
Chemnitz, Germany

REVIEWED BY
Afshin Shoeibi,
K. N. Toosi University of
Technology, Iran
M. Raheel Bhutta,
Sejong University, South Korea

*CORRESPONDENCE
Keum-Shik Hong
kshong@pusan.ac.kr

RECEIVED 10 February 2022
ACCEPTED 20 July 2022
PUBLISHED 31 August 2022

CITATION
Mughal NE, Khan MJ, Khalil K, Javed K, Sajid H, Naseer N, Ghafoor U and Hong K-S (2022) EEG-fNIRS-based hybrid image construction and classification using CNN-LSTM. *Front. Neurorobot.* 16:873239. doi: 10.3389/fnbot.2022.873239

EEG-fNIRS-based hybrid image construction and classification using CNN-LSTM

Nabeeha Ehsan Mughal[1], Muhammad Jawad Khan[1,2], Khurram Khalil[1], Kashif Javed[1], Hasan Sajid[1,2], Noman Naseer[3], Usman Ghafoor[4] and Keum-Shik Hong[4]*

[1]School of Mechanical and Manufacturing Engineering, National University of Sciences and Technology (NUST), Islamabad, Pakistan, [2]National Center of Artificial Intelligence (NCAI) – NUST, Islamabad, Pakistan, [3]Department of Mechatronics and Biomedical Engineering, Air University, Islamabad, Pakistan, [4]School of Mechanical Engineering, Pusan National University, Busan, South Korea

The constantly evolving human–machine interaction and advancement in sociotechnical systems have made it essential to analyze vital human factors such as mental workload, vigilance, fatigue, and stress by monitoring brain states for optimum performance and human safety. Similarly, brain signals have become paramount for rehabilitation and assistive purposes in fields such as brain–computer interface (BCI) and closed-loop neuromodulation for neurological disorders and motor disabilities. The complexity, non-stationary nature, and low signal-to-noise ratio of brain signals pose significant challenges for researchers to design robust and reliable BCI systems to accurately detect meaningful changes in brain states outside the laboratory environment. Different neuroimaging modalities are used in hybrid settings to enhance accuracy, increase control commands, and decrease the time required for brain activity detection. Functional near-infrared spectroscopy (fNIRS) and electroencephalography (EEG) measure the hemodynamic and electrical activity of the brain with a good spatial and temporal resolution, respectively. However, in hybrid settings, where both modalities enhance the output performance of BCI, their data compatibility due to the huge discrepancy between their sampling rate and the number of channels remains a challenge for real-time BCI applications. Traditional methods, such as downsampling and channel selection, result in important information loss while making both modalities compatible. In this study, we present a novel recurrence plot (RP)-based time-distributed convolutional neural network and long short-term memory (CNN-LSTM) algorithm for the integrated classification of fNIRS EEG for hybrid BCI applications. The acquired brain signals are first projected into a non-linear dimension with RPs and fed into the CNN to extract essential features without performing any downsampling. Then, LSTM is used to learn the chronological features and time-dependence relation to detect brain activity. The average accuracies achieved with the proposed model were 78.44% for fNIRS, 86.24% for EEG, and 88.41%

for hybrid EEG-fNIRS BCI. Moreover, the maximum accuracies achieved were 85.9, 88.1, and 92.4%, respectively. The results confirm the viability of the RP-based deep-learning algorithm for successful BCI systems.

KEYWORDS

recurrence plots (RP), convolutional neural networks (CNN), time distributional layers, long-short term memory (LSTM), brain computer interface (BCI)

Introduction

Brain–computer interfaces (BCIs) have become an indispensable element for individuals with disabilities. They have become integral components of new medical applications and have been increasingly applied in communication systems, human–machine interfaces (Bai et al., 2020), and neurofeedback applications (Mercado et al., 2021). BCI enables communication between the human brain and the external computer/device through generated brain commands, thereby avoiding the peripheral nervous system (Antonietti et al., 2021). Moreover, BCI is a neurofeedback method that can enhance the quality of life of patients suffering from serious motor debilities due to tetraplegia (Benaroch et al., 2021), stroke (Mane et al., 2020), and other spinal cord injuries (Al-Taleb et al., 2019). BCI also has applications in neurorehabilitation, communication and control, motor therapy and recovery, brain monitoring, and neuro-ergonomics (Asgher et al., 2020a,b; Mughal et al., 2021). The BCI analyzes a biosignal measured from a healthy subject to predict some intangible aspects of their cognitive state. This process usually consists of three main steps: data acquisition from the brain depending on the application and modality chosen, interpretation or pre-processing data into commands, and output to the computer to generate a command. Among the three types of BCI, namely, reactive, active, and passive BCI (pBCI), pBCI is an important research area that estimates human emotions, cognition, intentions, and behavior based on generated brain responses to different situations.

The demand for improved traditional BCI practices has increased with advances in neuroimaging modalities. Primary non-invasive neuroimaging modalities for BCI include functional magnetic resonance imaging (fMRI), electroencephalography (EEG), magnetoencephalography, and functional near-infrared spectroscopy (fNIRS). Among them, EEG and fNIRS are the foremost modalities in terms of cost and manageability (Rahman et al., 2020; Rashid et al., 2020). EEG measures brain activity by calculating the voltage fluctuations from the action potentials of neurons, whereas fNIRS detects brain activity related to hemodynamic response changes (Hong and Zafar, 2018; Liu et al., 2021). Although invasive techniques provide more accurate data than non-invasive techniques, non-invasive modalities are more frequent and appreciated in the research domain. Non-invasive recording techniques for brain activity improve safety and reduce ethical concerns (Burwell et al., 2017; Pham et al., 2018). Over time, various non-invasive techniques have been used in studies. The most commonly used are EEG, fNIRS, electrooculography, and fMRI (Choi et al., 2017). The selection of a non-invasive modality depends on many factors. Usually, the following parameters are considered: cost, ease of use, and temporal and spatial resolution, as needed by the application. Each modality offers some advantages over the others, and there are always some associated trade-offs; the pros of one modality compensate for the cons of the other modality. Thus, hybrid approaches have proven to be more efficient. Hybrid neuroimaging modalities increase accuracy and offer a greater degree of reasonable control (Hong and Khan, 2017; Khan and Hong, 2017; Hong et al., 2018).

Researchers appreciate the use of low-cost neuroimaging modalities (Hong et al., 2020). Modalities that offer convenience for non-laboratory setups are also choices of interest. In this regard, EEG and fNIRS are the most commonly used. Both are portable and inexpensive compared to the alternatives. Electrodes capture EEG signals due to variations in the current generated by neurons due to postsynaptic activities (Sazgar and Young, 2019). Several electrodes are placed on the subject's scalp for EEG data acquisition. Although EEG provides better temporal resolution ranging up to ∼0.05 s, it provides a spatial resolution of only ∼10 mm (Puce and Hämäläinen, 2017; Fu et al., 2020). The contrasting comparison of the temporal and spatial resolutions manifests trade-offs when using the EEG modality. In contrast to EEG, fNIRS is an optical imaging technique that measures light absorbance to calculate concentration changes in oxy-hemoglobin and deoxy-hemoglobin within the brain. Similar to EEG, fNIRS is cost-effective and portable. However, unlike EEG, fNIRS provides better spatial resolution. Moreover, fNIRS is less influenced by electrical noise (Hasan et al., 2020; Ghafoor et al., 2022). As evidenced by the comparison, fNIRS can compensate for the trade-offs of EEG. Thus, the EEG and fNIRS hybrid method serve as a breakthrough in neuroimaging (Ahn and Jun, 2017) on theoretical grounds.

As fNIRS measures hemodynamic responses, there is an innate delay in the measurement (Saeed et al., 2020).

Various methods have been proposed to compensate for this slow command generation. In this regard, a hybrid method comprising EEG and fNIRS techniques can be used, which proceeds by measuring the initial dip [i.e., at the onset of neural firing, the oxygenated hemoglobin (HBO) level first decreases] instead of the actual hemodynamic response (Hong and Khan, 2017; Kamran et al., 2018). The other contrasting difference between the two modalities is the rate at which data are sampled. The EEG data acquisition rate is ∼10–100 times faster than that of fNIRS. When the EEG and fNIRS hybrid is used, it is common to downsample the EEG data to make its processing compatible with that of the fNIRS data (Khan and Hasan, 2020; Ortega et al., 2020). Downsampling might discard some segments of valuable data. As EEG signals are prone to electrical noise, fNIRS suffer from physiological noise, instrumentation, and experimental errors. The experimental errors may be spontaneous, unintentional diversions from the intended protocol, such as motion artifacts or changes in the light intensity in the ambiance. The motion artifacts present in the data can be significantly reduced via Wiener filtering-based methods (Jiang et al., 2019) or wavelet analysis-based methods (Islam et al., 2021). Instrumentation can also induce noise in the data, such as noise from the hardware. However, these noise signals are high-frequency components; thus, they can be eliminated using a low-pass filter. Physiological noises can arise due to breathing activity or heartbeats. Although these noises are unavoidable, many methods have been reported to counter these noises; commonly used techniques apply bandpass filters, parameter mapping, and independent component analysis (Rejer and Cieszyński, 2019; Vourvopoulos et al., 2019; Wankhade and Chorage, 2021). Denoising the data further removes data regions; thus, the processed data are even smaller in magnitude than the raw data. Therefore, the downsampling of the EEG data after pre-processing to match the fNIRS data removes a considerable amount of valuable information regarding brain activity.

Recurrence quantification analysis (RQA) of RP has become popular in recent years for analyzing brain activity because brain signals are both recurrent and dynamic. RP, in general terms, is a non-linear evaluation method for recurrent and dynamic signals. It is a visualization displaying the recurrent occurrences of states $x(n)$ of a time signal in phase space. RQA is an analysis technique used to quantify the features of the constructed RP. In the literature, RQA feature analysis has been used in EEG signal detection of epilepsy and Alzheimer's disease, coupling and synchronization in EEG of epileptic discharge, and so on. Cortical function during different sleep stages was also analyzed using RP features. The RQA analysis showed that unique RPs were extracted for different sleep stages (Parro and Valdo, 2018). Several studies have also used artificial neural networks (ANNs) (Torse et al., 2019) and support vector machines (SVMs) (Houshyarifar and Amirani, 2017; Zhao et al., 2021) to classify extracted RQA features. One study used a four-layer ANN for different EEG channels to predict the onset of seizures using RQA measures (Torse et al., 2019).

As machine learning (ML) has rapidly become a state-of-the-art analysis tool, researchers have considered searching for classification features (Park and Jung, 2021). The qualitative aspects of these RPs can be used for classification. Moreover, DNNs are highly efficient training classifiers, resulting in better classification accuracy than ML classifiers (Sattar et al., 2021). However, only a few studies that applied these algorithms in BCI are available (Dehghani et al., 2021; Singh et al., 2021). Only one study used a CNN for the binary classification of epileptic seizures from EEG using RP as images (Gao et al., 2020). The practical application of biological feedback in BCI requires efficient and precise motor activity detection and classification methods. These conventional quantification and feature selection methods and simple ML classifiers face several challenges when implementing real-time BCI. Traditional feature engineering methods involve multiple steps, such as feature extraction, feature selection, finding suitable combinations for various features, and sometimes dimensionality reduction from a comparatively small quantity of data, thus leading to other problems such as overfitting and bias (Asgher et al., 2020b). These inherent constraints hinder adjustments by researchers. Therefore, the initial analysis steps, namely, data mining and pre-processing, are time-consuming.

On the other hand, deep learning (DL) algorithms such as CNNs can be employed in two ways for BCI applications: altering or modifying the CNN algorithm architecture to accommodate the one-dimensional time-series data obtained by the modalities or transforming one-dimensional data into two-dimensional (2D) data to be conveniently input to the CNN. Deep neural networks (DNNs) and other traditional classifiers have also been employed based on fNIRS and EEG signals to recognize three different cognitive states (Huve et al., 2019; Takahashi et al., 2021), electromyography signals classification (Oh and Jo, 2021), control of wearable exoskeleton (Sun et al., 2021), and other control applications (Kim et al., 2021; Li et al., 2021; Yaqub et al., 2021). A similar approach has been used for various other applications, such as controlling robots (Huve et al., 2018), differentiating workloads by analyzing the fNIRS signals, and using deep learning techniques. Shoeibi et al. (2022) used an adaptive neuro-fuzzy interface to detect epileptic seizures from EEG signals. The literature also demonstrates the time-delay neural network for classification purposes. Thyagachandran et al. (2020) used this approach to classify EEG signals; however, the presented model was not sufficiently deep to learn the hierarchical features of the EEG signal. The research that resonated most with our present study is that of Tanveer et al. (2019). The authors investigated deep learning-based BCI to detect driver drowsiness. The output strength of the selected channels was translated into color maps and fed into the CNN classifier as an input. The output color maps were obtained by linear mapping the values from the channel to the color intensity. Recent biosignal analysis techniques have a higher inclination toward non-linear dynamics. One of the most widely used methods is the recurrence plot (RP). The analysis focuses on the repeatability of the time-series states and presents the output in geometric structures, whose topology is analyzed to estimate the characteristics of the dynamics (Nayak et al., 2018; Acharya et al., 2019). In the literature, researchers have also experimented with hybrid CNN-LSTM models for time-series biological signal analysis to detect mental disorders. A study to detect schizophrenia via EEG was carried out by Shoeibi et al. (2021) using different ML and DL models, and a comparison was made between applied algorithms based on their accuracy percentage. Among all ML and DL algorithms, CNN-LSTM proved the best architecture for diagnosing schizophrenia.

This study investigated the RP performance for EEG, fNIRS, and hybrid EEG-fNIRS within a deep convolutional neural network and a long short-term memory (CNN-LSTM) model for neuroimaging brain data for BCI. The obtained RPs of EEG and fNIRS were fed as images into the hybrid CNN-LSTM network for classification. The initial hypothesis of this study was that "The classification accuracy of Hybrid EEG-fNIRS BCI will improve by incorporating all signal information from both modalities using recurrence plots instead of using traditional methods of downsampling EEG signals to make them compatible with fNIRS for hybrid BCI." The main contributions and novelty of our work are as follows:

(i) Implementing whole EEG and fNIRS signals, without any information loss or downsampling, for Hybrid EEG-fNIRS BCI using recurrence plots.
(ii) Implement the time-distributed CNN-LSTM model for activity detection using EEG and fNIRS recurrence plots for hybrid BCI.

Furthermore, to the best of the authors' knowledge, time-distributional (TD) layers were implemented in a network that was not previously used in the BCI field.

The detailed methodology of this research, the dataset used, RP formation from EEG and fNIRS datasets, and the classification approach used for the four-class classification of constructed RP are detailed in the following sections. The related performance of RP in EEG-BCI, fNIRS-BCI, and hybrid EEG-fNIRS-BCI is discussed, and the study's conclusions are provided.

Methodology

In this study, RP performance for EEG, fNIRS, and hybrid EEG-fNIRS with the deep CNN-LSTM model was investigated for neuroimaging brain data for BCI. RP transformed the time series data into the phase space and provided an alternate method to envisage the periodic nature of a time series trajectory, that is, brain signal data in phase space. RPs of EEG and fNIRS were constructed and used as images to feed the hybrid time-distributed CNN-LSTM network for classification. The detailed methodology is described in this section and illustrated in Figure 1.

Dataset and experimental protocol

The research used an open-source meta-dataset. The data were recorded at the Technische Universität Berlin (Shin et al., 2018) and collected through three different paradigms from 26 healthy participants while focusing on cognitive tasks. Datasets A, B, and C were chosen for the three different cognitive tasks: *n*-back, discrimination response, and word generation, respectively. On these grounds, the selected dataset was an appropriate choice for research in the domain of hybrid BCI. First, task A was performed, followed by tasks C and B. In this study, only dataset A (*n*-back) was used. The entire n-back dataset consisted of three sessions, and each session consisted of three series: 0-back, 2-back, and 3-back tasks. The total recording time for each series was 62 s. The initial 2 s were dedicated to task illustration. The following 40 s were reserved for the task performance (20 numbers displayed for 2 s each on the screen), and the last 20 s were reserved for the rest period. Thus, for every n-task, there were 180 trials. The experimental protocol for the n-back dataset is shown in Figure 2.

Data acquisition

The EEG and fNIRS data were recorded simultaneously to ensure that the data were synchronized, and a parallel port was used to send the triggers. A BrainAmp EEG amplifier was used to record the EEG data, and the sampling frequency was 1,000 Hz. A stretchable fabric cap was used to place the 30 active electrodes to acquire data in frontal, motor cortex, parietal, and occipital regions according to the internationally recognized 10–5 system (Shin et al., 2018).

The fNIRS data were recorded at a sampling frequency of 10.4 Hz via NIRScout (NIRx Medizintechnik GmbH, Berlin, Germany). Sixteen electrodes, representing a combination of sources with detectors, were positioned at the frontal lobe, motor cortex, parietal lobe, and occipital lobe. The optodes of the NIRS were fixed with EEG electrodes on the same cap. The positioning of the electrodes and optodes is illustrated in Figure 3 green circles: EEG electrodes, red circles: NIRS optodes.

Data pre-processing and labeling

The EEG data were downsampled to 200 Hz. A 6th-order Butterworth bandpass filter with a passband frequency range

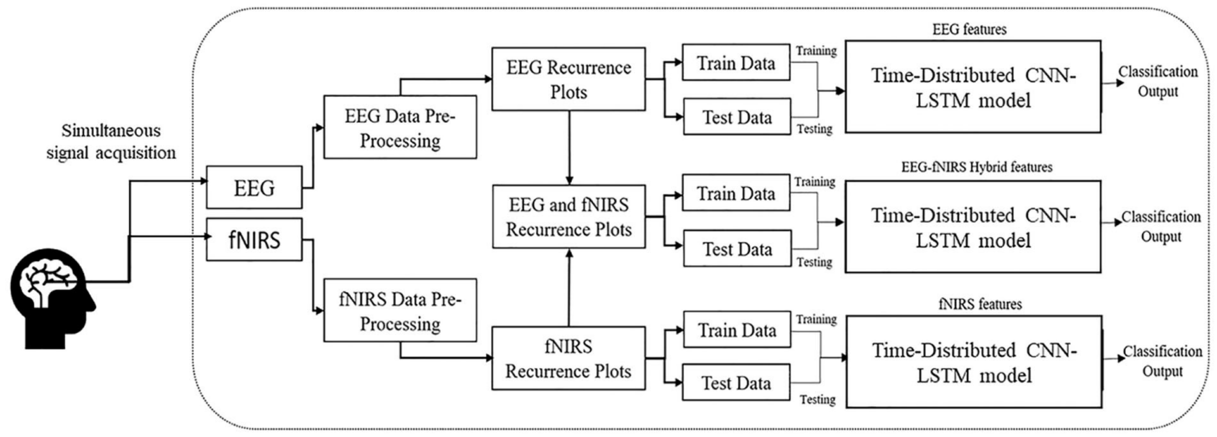


FIGURE 1
Methodology of the study shows the construction of the hybrid EEG-fNIRS dataset using RPs and classification using time-distributed CNN-LSTM.

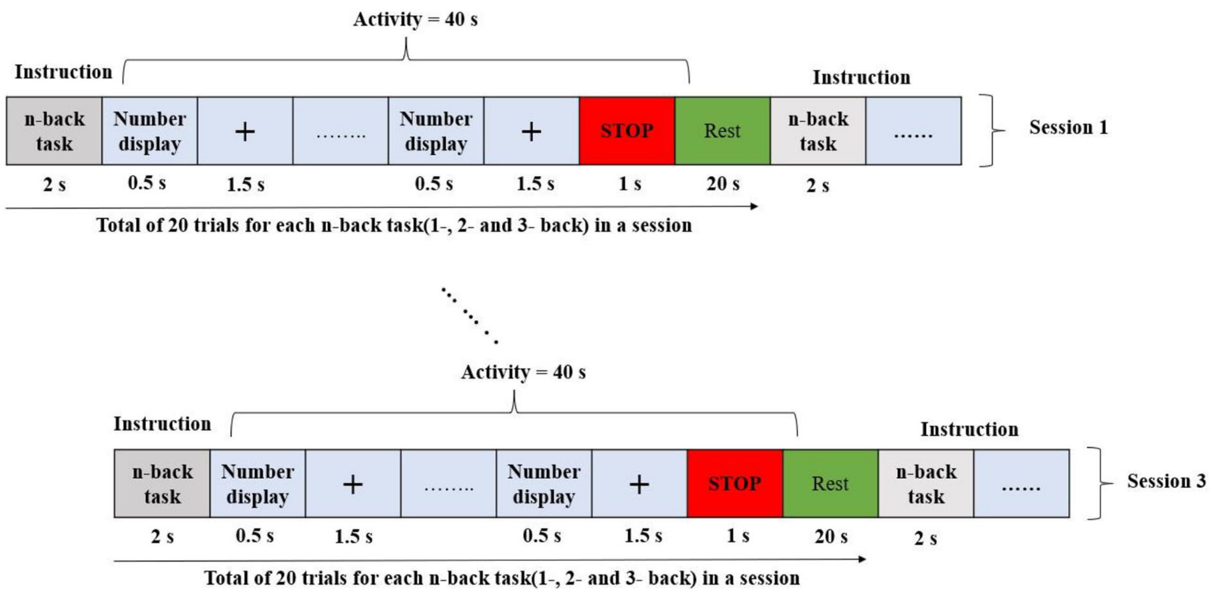


FIGURE 2
Experiment paradigm of n-back tasks.

of 1–40 Hz was used for filtering purposes. The acquired data were first translated into oxy- and deoxy-hemoglobin intensity variations to pre-process the fNIRS data. The conversions were conducted using the modified Beer–Lambert law. The fNIRS raw data were downsampled at 10 Hz. As the fundamental frequency of this dataset was very low, the downsampled data were not fed into the Butterworth bandpass filter. Instead, the data were low-pass filtered to avoid losing the fundamental frequency component. The cutoff frequency of the filter was chosen to be 0.2 Hz. The data were acquired using MATLAB R2013b software. Further processing was performed using Python on Spyder in the Anaconda development environment. After filtration, the dataset was labeled using the activity time markers from the acquired continuous EEG and fNIRS signals. Four classes, namely, 0-, 2-, and 3-back classes, and one of the remaining states, were labeled concerning the experimental protocol. After that, the labeled data were used for RP construction.

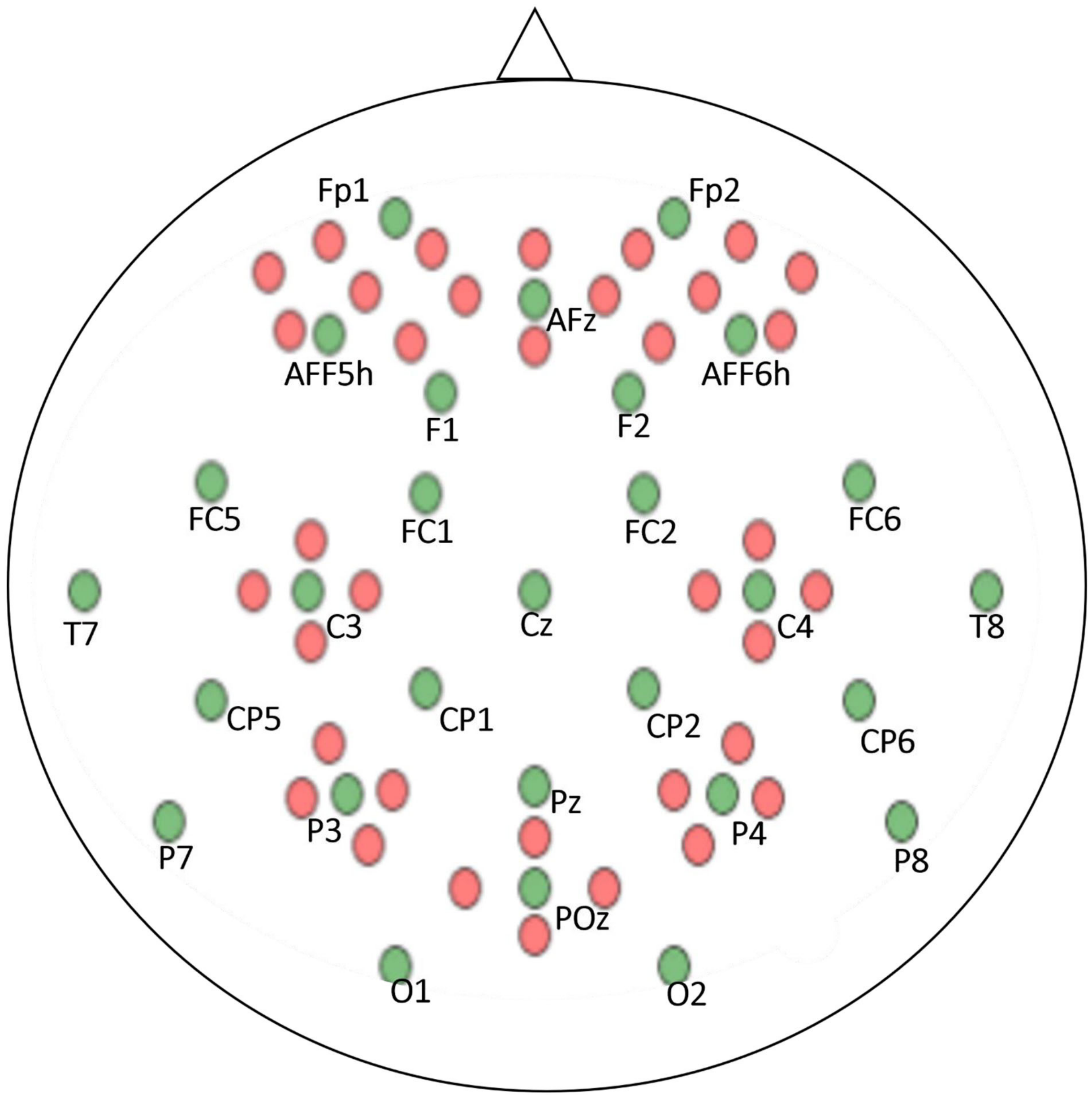


FIGURE 3
EEG and NIRS electrode positions according to the 10–5 system. Green dots represent EEG electrodes and red dots denote NIRS Optodes.

Recurrence plots

A recurrence plot (RP) is a contemporary technique for analyzing non-linear data. This technique employs the visualization of a square matrix whose elements link to the dynamic state repetition. The ordered pair of matrices corresponds to the specific timing of the repetition. Recurrence analysis is a graphical technique that aims to identify hidden recurring patterns (Ledesma-Ramirez et al., 2020). To illustrate this idea, our desired information is univariate time series data and that the data under analysis are a subpart of the large n-dimensional dataset. The topological rendering of the original n-dimensional dataset can be obtained using a single observable variable.

Thus, the embedded matrix, namely, x^{m}, can be constructed as follows:

$$[x_i^m = (x_i, x_{i+d}, x_{i+2d}, \ldots\ldots, x_{i+(m-1)d})] \tag{1}$$

where x_i is a scalar series, the dimension is represented by m, and d is the delay. In case the condition

$$m \geq 2n + 1 \tag{2}$$

is satisfied, the single output variable exhibits the potential to recreate the entire system. Recreation heavily depends on the sequence of the embedded matrix. The sequence can be controlled by adequately choosing parameters m and d. The asymmetric matrix of the Euclidean distances can also be constructed by measuring the distance between pairs of embedded vectors. These distances are translated into an equivalent color, and each distance has a distinctive color. Thus, an RP is a square assortment of pixels whose color depends on the corresponding magnitude of values. The pixel coordinates also carry useful information that is linearly related to the location of the data in the original data matrix.

The use of ε is commonly employed in RPs. This ε is referred to as the critical radius. Each value is compared with the critical radius to check whether the pixel value is $\leq\varepsilon$; then, the pixel is displayed as a darkened pixel. In other words, RP is a visualization of a square recurrence matrix showing all the instances of times at which a state of a non-linear system repeats; the columns and axes of the recurrence matrix correspond to specific time intervals. In technical terms, an RP shows every time of a non-linear time signal from a dynamical system at which its phase space trajectory spans approximately the same area in the phase space. In graphical terms, this is a graph of

$$\vec{x}(i) \approx \vec{x}(j) \tag{3}$$

where i is on the horizontal axis, j is on the vertical axis, and $\vec{x}$ is the phase space trajectory of the dynamical system. Thus, a binary recurrence matrix is constructed using a specific time window $w = 5$ s, where any two-time steps are separated by the time interval $\varepsilon = 0.1$ and a step size of 10 in the following manner:

$$R(i,j) = \begin{cases} 1 \ \ if \ \left\| \vec{x}(i) - \vec{x}(j) \right\| \ \leq \ \varepsilon \\ 0 \ \ otherwise \end{cases} \tag{4}$$

where i and j are the horizontal and vertical time axes, $i, j \in \{t_0, t_1 \ldots .t, \ldots . t_T\}$. The RP is a visualization of the recurrence matrix with a black square of the lattice at coordinates (i, j) if $R(i, j) = 1$ and a white square if $R(i, j) = 0$.

Figure 4 shows the corresponding RPs constructed using the fNIRS dataset for the 0-, 2-, and 3-back classes, and the rest state for Subject 1.

After experimenting with different values for the parameters of the RPs, $\varepsilon = 0.1$ and step size $= 10$ were adopted. Figure 4 depicts the non-linear mapping of the acquired brain signals to the new dimension through the RPs with the selected parameters. Each subject's RP data were split into training and test datasets using 10-fold cross-validation before performing classification to avoid overfitting and provide better generalization. Moreover, the model performance was evaluated based on the following performance metrics.

Accuracy

The accuracy of a classifier is the proportion of the total number of correct predictions made by the classifier. If the confusion matrix is given, accuracy is defined as:

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Number\ of\ Samples}$$

Precision

Precision or positive predictive value is the proportion of the correctly predicted positive cases of all cases predicted positively by a classifier. Given a confusion matrix, precision is defined as:

$$Precision = \frac{True\ Positives}{Total\ Number\ of\ Samples\ Predicted\ as\ Positive}$$

Recall

Recall or sensitivity is the proportion of all actual positive cases that were correctly predicted to be positive by a classifier. Given a confusion matrix, recall is defined as:

$$Recall = \frac{True\ Positives}{Total\ Number\ of\ Actual\ Positive\ Samples}$$

F1 score

The F1 score is the harmonic mean of precision and recall values for a given classification model. Given a confusion matrix, the F1 score is defined as:

$$Precision = \left(\frac{Precision^{-1} + Recall^{-1}}{2} \right)^{-1}$$

Time distributed CNN-LSTM

A CNN is a multilayered neural network with architecture to detect complex features in the data. Unlike traditional multilayer perceptron architectures, CNN uses two operations called "convolution" and "pooling" to reduce the image into its essential features, which are used to understand and classify it. CNNs are composed of basic building blocks, which

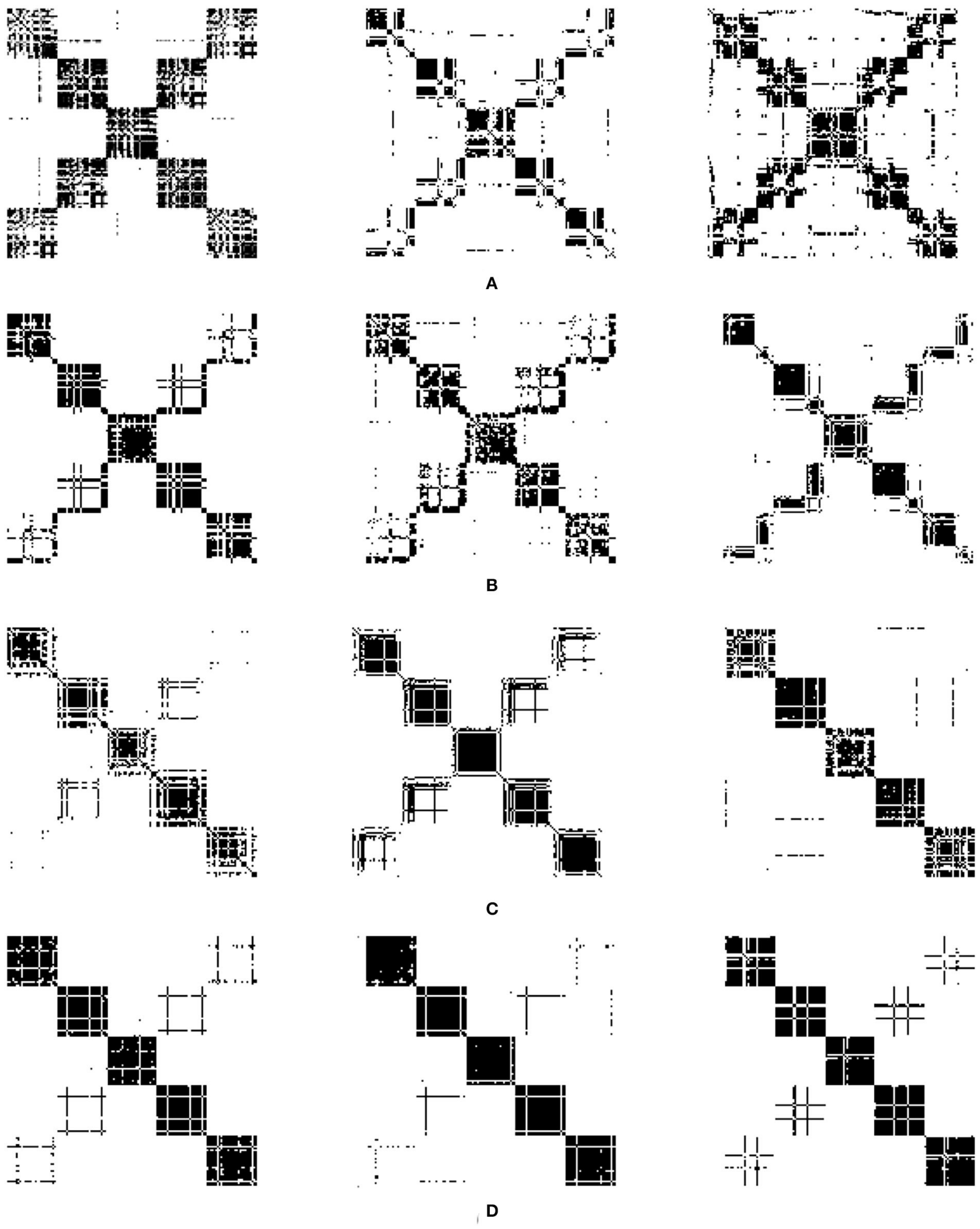


FIGURE 4
RPs of fNIRS dataset. **(A)** 0-back, **(B)** 2-back, **(C)** 3-back, **(D)** rest.

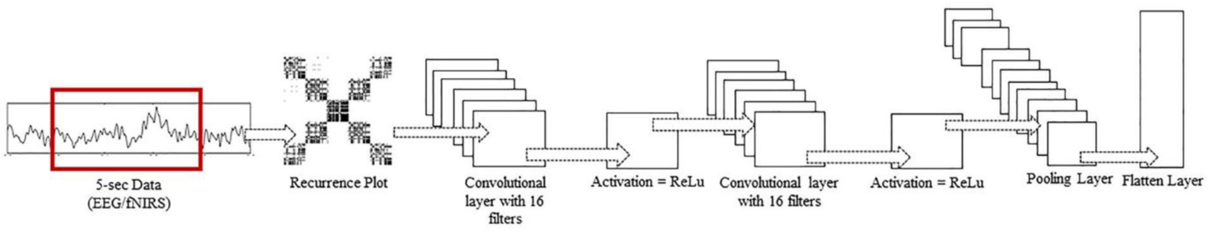


FIGURE 5
Inside a TD layer. RP input to two Conv2D layers, each with 16 filters, and ReLu as activation function, followed by a max pool and flatten layer.

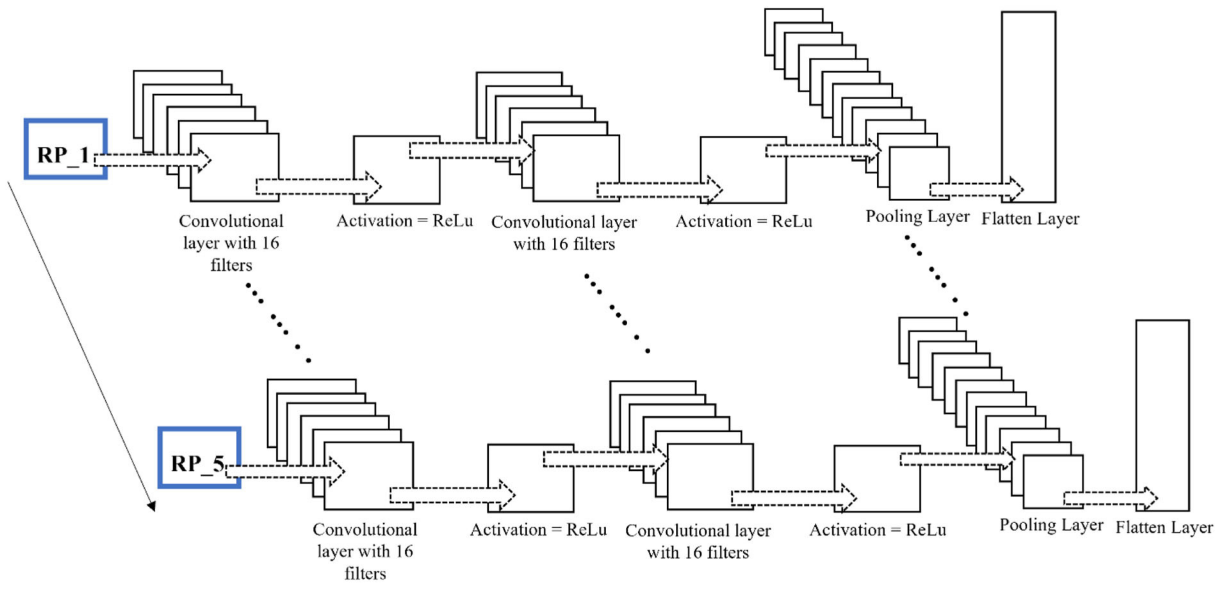

RP_1
Convolutional layer with 16 filters
Activation = ReLu
Convolutional layer with 16 filters
Activation = ReLu
Pooling Layer
Flatten Layer
RP_5
Convolutional layer with 16 filters
Activation = ReLu
Convolutional layer with 16 filters
Activation = ReLu
Pooling Layer
Flatten Layer

FIGURE 6
Time-distributed CNN-LSTM network for classification of four-class mental workload using RPs of EEG and fNIRS dataset.

include the following: a convolutional layer with a filter or kernel passed over an image; an activation layer that usually has an activation function; a rectified linear unit (ReLU) to introduce non-linearity that allows the network to train itself through backpropagation; a pooling layer that downsamples and reduces the size of the matrix and is focused on the most prominent information in each feature of the image; and a fully connected layer that outputs the different probabilities associated with every label attached to the image. The label with the highest probability is the classification decision. CNNs are widely used in agriculture, self-driving vehicles, healthcare, and surveillance. LSTM networks are recurrent neural networks (RNNs) that use special and standard units. The special units include the "memory cell," which maintains information in its memory for longer. LSTM has feedback connections, unlike standard feed-forward neural networks; it can process entire data sequences, including speech and video. LSTM is widely used in speech recognition, handwriting recognition, handwriting generation, music generation, language translation, image captioning, and anomaly detection in intrusion detection systems. A simple LSTM unit comprises a cell, input, output, and forget gate. The cell remembers the information, whereas the gates regulate the flow of information. LSTM networks are modified forms of RNNs; they remember past data in memory.

Over time, researchers have applied different architectures and types of deep learning networks. Unlike images, text, voice, and other widely used datasets, neuroimaging signals are intrinsically different and have an important chronological order. This chronological order dictates the flow of information necessary to detect activities or actions. Examples of such chronological order are the initial dip at the start of activity in

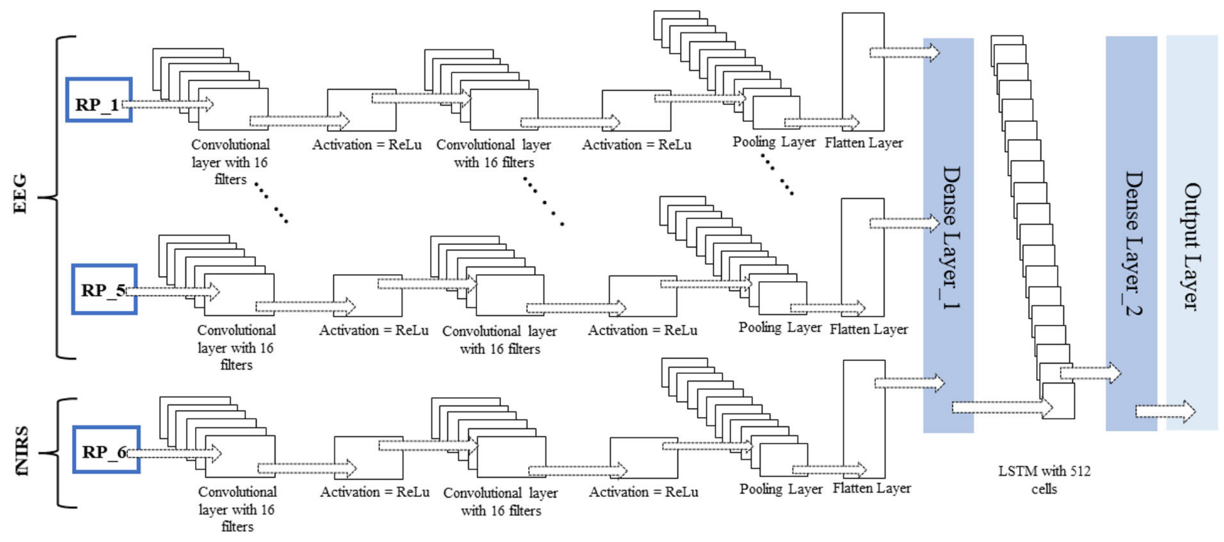


FIGURE 7
Time-distributed CNN-LSTM network for classification of four-class mental workload using RPs of hybrid EEG-fNIRS dataset.

fNIRS signals and positive deflection in event-related potential P300 signals in EEG. A novel CNN-LSTM network was designed for this study. The network consists of one CNN and one LSTM module combined with a dense layer. After pre-processing, the data are fed into the CNN module; this module consists of two convolutional layers, each with 16 filters and ReLU as the activation function, and one max-pooling layer. CNNs are best known for their abilities of feature extraction from 2D and 3D images. Considering the data sequence used in the form of chronologically ordered time windows, the relationship between two windows in a given input should be detected. An LSTM layer enables the network to use its memory and enhance its prediction. The convoluted output from the CNN block is reshaped and flattened before being fed into the LSTM layer. The layers preceding the LSTM layers are wrapped inside a time-distributed layer that allows their application to every temporal slice of the input data. This time-distributed wrapper applies the same instance of convolutional layers to each timestamp, such that the same set of weights is used. After passing through another dense layer, the LSTM layer terminates into the output layer.

No researcher has exploited this chronological order using time-distributed layers in deep learning models to the authors' knowledge. The constructed RPs with a fixed window length and an overlapping portion are fed into the network as images. The different configurations of this proposed network for fNIRS, EEG, and hybrid modalities are discussed in detail in the Discussion section. The network architecture for the EEG and fNIRS BCI and the details of the hyperparameters of the DL model used, that is, several layers, dimensions, the number of filters used in each layer, and the number of neurons, among other details, are shown in Figure 7. Researchers have invested tremendous efforts to determine the single best architecture for deep learning neural networks, giving rise to the sub-research field known as neural architecture search (NAS). However, there is no definite answer regarding the optimal neural architecture a priori. The number of neurons, number of filters, number of layers, their combinations, dropout, and max-pooling percentage remain the best hyperparameters. The most viable approach seems to be using intuition and domain knowledge to determine an initial guess for these parameters and then iteratively shortlist them to obtain good values. In this study, the NAS design process was as follows: a network with a minimum number of parameters, a single convolutional layer, a single LSTM layer, and one dense layer was created; other hyperparameters were tuned; more layers were added, and the network hyperparameters were tuned with a grid search using the sklearn wrapper. We performed the above grid search with sample data and chose the best-performing network for EEG, fNIRS, and EEG+fNIRS datasets. However, this approach resulted in input dimension mismatch because of the extra number of features in the hybrid dataset compared to the single modality datasets. We solved this problem by adding another sequence module on top of the EEG network architecture and wrapping it inside the TD layer, similar to the EEG network. The later stages of a network-like dense layer, LSTM layer, and the following layers remained the same; however, this strategy solved the input dimensionality mismatch problem. Figures 5–7 show the network architecture of the study.

TABLE 1 Performance evaluation metrics of fNIRS-BCI, EEG-BCI, and hybrid EEG-fNIRS-BCI.

	S1				S2			
	Accuracy	**Precision**	**Recall**	**F-1 score**	**Accuracy**	**Precision**	**Recall**	**F-1 score**
fNIRS-BCI	81.32	81.00	80.66	80.53	81.12	82.20	80.35	80.72
EEG-BCI	85.11	85.41	84.87	84.66	86.34	86.44	86.01	86.00
Hybrid-BCI	89.63	90.09	89.26	89.45	86.78	87.19	86.13	85.90
	S3				**S4**			
	Accuracy	**Precision**	**Recall**	**F-1 score**	**Accuracy**	**Precision**	**Recall**	**F-1 score**
fNIRS-BCI	77.25	77.80	76.56	76.59	71.74	72.15	70.70	70.52
EEG-BCI	86.53	87.01	86.15	86.11	85.60	85.87	84.76	84.47
Hybrid-BCI	88.33	88.41	88.09	87.91	86.66	86.88	86.61	86.58
	S5				**S6**			
	Accuracy	**Precision**	**Recall**	**F-1 score**	**Accuracy**	**Precision**	**Recall**	**F-1 score**
fNIRS-BCI	70.57	71.26	69.75	69.82	76.74	78.70	75.79	76.08
EEG-BCI	89.31	89.33	89.05	89.01	85.55	85.22	84.92	84.76
Hybrid-BCI	91.79	92.01	91.36	91.37	89.81	89.57	89.39	89.28
	S7				**S8**			
	Accuracy	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 score**
fNIRS-BCI	77.74	78.53	77.28	76.90	82.44	82.82	82.15	82.10
EEG-BCI	89.00	89.44	88.79	88.84	82.14	82.53	81.89	81.98
Hybrid-BCI	92.28	92.57	92.27	92.28	83.32	83.15	82.76	82.35
	S9				**S10**			
	Accuracy	**Precision**	**Recall**	**F-1 score**	**Accuracy**	**Precision**	**Recall**	**F-1 score**
fNIRS-BCI	80.40	80.98	79.83	79.84	80.14	80.38	79.52	79.59
EEG-BCI	88.76	89.25	88.61	88.51	84.06	84.70	84.00	84.01
Hybrid-BCI	90.62	90.61	90.28	90.38	87.40	87.90	87.45	87.25
	S11				**S12**			
	Accuracy	**Precision**	**Recall**	**F-1 Score**	**Accuracy**	**Precision**	**Recall**	**F-1 score**
fNIRS-BCI	79.16	79.67	78.34	78.27	77.18	76.65	76.37	75.85
EEG-BCI	83.93	84.09	83.50	83.53	89.75	89.89	89.56	89.59
Hybrid-BCI	86.71	86.88	86.54	86.51	91.35	91.45	91.18	91.13
	S13				**S14**			
	Accuracy	**Precision**	**Recall**	**F-1 score**	**Accuracy**	**Precision**	**Recall**	**F-1 score**
fNIRS-BCI	75.15	75.39	74.22	74.16	77.00	79.12	75.66	76.23
EEG-BCI	89.63	89.92	89.10	88.51	87.84	87.57	87.36	87.26
Hybrid-BCI	91.91	92.11	91.73	91.68	89.93	89.90	89.72	89.57

(Continued)

TABLE 1 Continued

	S15				S16			
	Accuracy	Precision	Recall	F-1 score	Accuracy	Precision	Recall	F-1 score
fNIRS-BCI	74.64	75.56	73.45	73.47	78.66	79.20	77.80	77.71
EEG-BCI	86.35	87.24	85.97	85.51	82.33	82.61	81.48	81.59
Hybrid-BCI	87.34	87.46	86.76	86.64	83.50	83.59	83.33	82.99

	S17				S18			
	Accuracy	Precision	Recall	F-1 score	Accuracy	Precision	Recall	F-1 score
fNIRS-BCI	80.27	80.60	79.60	79.75	81.76	82.07	81.48	81.48
EEG-BCI	87.77	87.88	87.78	87.60	89.68	89.92	89.62	89.49
Hybrid-BCI	92.16	92.83	91.69	91.81	93.58	93.67	93.66	93.56

	S19				S20			
	Accuracy	Precision	Recall	F-1 Score	Accuracy	Precision	Recall	f-1 Score
fNIRS-BCI	79.35	80.48	79.55	79.17	80.33	80.86	79.49	79.71
EEG-BCI	86.16	86.08	85.69	85.64	87.52	87.80	86.92	87.01
Hybrid-BCI	87.63	87.84	86.91	87.04	91.36	91.87	91.06	91.11

	S21				S22			
	Accuracy	Precision	Recall	F-1 Score	Accuracy	Precision	Recall	F-1 Score
fNIRS-BCI	74.21	75.29	72.88	73.14	82.37	83.42	81.60	81.96
EEG-BCI	87.89	88.04	87.64	87.46	79.91	80.76	78.72	78.81
Hybrid-BCI	89.75	89.95	89.44	89.33	81.65	82.46	81.05	81.11

	S23				S24			
	Accuracy	Precision	Recall	f-1 score	Accuracy	Precision	Recall	F-1 score
fNIRS-BCI	79.60	79.75	78.90	78.91	78.61	78.82	78.05	78.12
EEG-BCI	84.37	83.94	83.70	83.62	86.96	87.64	86.52	86.77
Hybrid-BCI	85.36	85.18	84.73	84.46	89.81	90.03	89.70	89.62

	S25				S26			
	Accuracy	Precision	Recall	F-1 score	Accuracy	Precision	Recall	F-1 score
fNIRS-BCI	80.27	80.77	79.40	79.47	81.45	81.72	80.56	80.58
EEG-BCI	84.43	85.21	83.81	83.86	85.17	85.15	84.18	84.30
Hybrid-BCI	84.92	85.63	84.57	84.52	85.17	85.62	84.48	84.16

Results

The time-distributed CNN-LSTM was used in this research to classify four classes, namely, the three n-back activities and the rest state from the fNIRS dataset acquired from 26 subjects. The data acquisition and initial pre-processing included filtering EEG data using a zero-phase, low-pass, 6th-order Butterworth filter. In the case of fNIRS, conversion of light densities into changes in the concentrations of HbO and HbR (hemodynamic response) was performed using the modified Beer–Lambert law followed by a zero-phase, low-pass, 6th-order Butterworth filter. After that, the data were

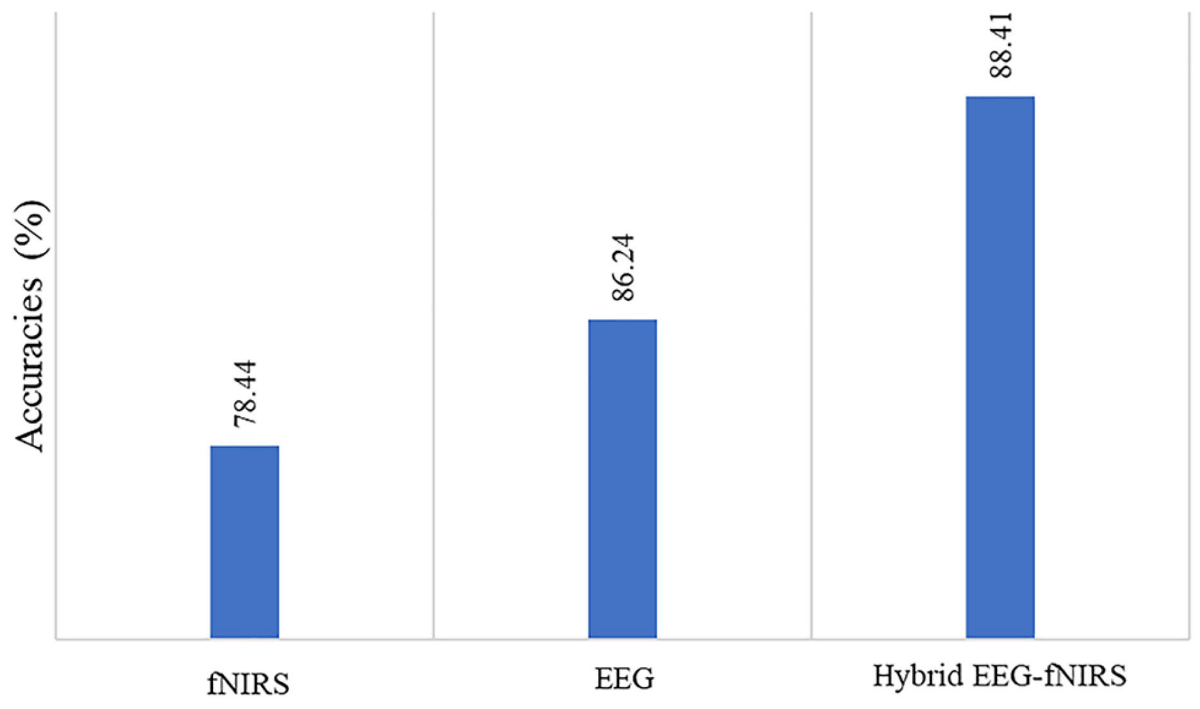


FIGURE 8
Comparison of accuracies of fNIRS-based BCI, EEG-based BCI, and hybrid EEG-fNIRS-based BCI.

labeled along with outlier rejection and data normalization. The window size selection in the hybrid EEG/fNIRS-based BCI system is essential because hemodynamics response has an inherent delay, which requires 0–10 s to complete after the stimulus. In the literature, researchers have experimented with different windows of varying lengths, such as 2–9, 2–7, and 3–7 s (Khan and Hong, 2017; Gaur et al., 2021). Generally, the smaller the window size, the better the BCI performance will be. After an initial investigation, a window size of 5 s with 20% overlap was used for all BCIs. After that, the RPs for segmented signals were constructed into a sequence of 5 s windows.

The deep learning algorithms were trained on a GTX 1060 graphic card with 3 GB VRAM and an Intel 6th Gen Core i7-6700HQ processor with a 3.2-GHz frequency. The Keras API was used with the TensorFlow backend on Spyder in the Anaconda integrated development environment. The average accuracy achieved for the four-class classification was 78.4% for fNIRS, 86.44% for EEG, and 88.41% for hybrid EEG-fNIRS BCI. The maximum accuracies achieved were 82.4, 89.75, and 93.58%, respectively. Table 1 summarizes the results of the 26 participants in terms of their classification accuracies, precision, recall, and F1 score for EEG-BCI, fNIRS-BCI, and Hybrid EEG-fNIRS BCI. Figure 8 shows the average accuracies achieved by these approaches.

Discussion

Researchers appreciate the use of low-cost neuroimaging modalities. Modalities that offer convenience to non-laboratory setups are also choices of interest. In this regard, EEG and fNIRS are the most commonly used neuroimaging modalities. Both are portable and inexpensive compared to fMRI. However, EEG offers a spatial resolution of only ∼10 mm (Puce and Hämäläinen, 2017; Fu et al., 2020). The contrasting comparison of the temporal and spatial resolutions manifests trade-offs when using the EEG modality.

In contrast to EEG, fNIRS constructs functional neuroimages of the brain by employing NIR light. As fNIRS measures hemodynamic responses, there is an innate delay in the measurement (Saeed et al., 2020). Various methods have been proposed to compensate for this slow command generation. In this regard, hybrid EEG-fNIRS can be an option. However, the sampling frequencies of both modalities are different, thus resulting in information loss. Moreover, the most important objective of all studies conducted on BCI is to enhance real-time classification accuracy and reduce computational costs with multiple commands, thus emphasizing the need to develop appropriate identification and classification methods for real-time BCI (Phanikrishna et al., 2021). Usually, multi-channel brain signal acquisition modalities (i.e., EEG) analyze brain motor activity using different methods,

TABLE 2 Comparison with other 4-class classification studies for BCI.

Authors	Brain acquisition modality	Classes	Subjects	Methods	Performance %
Ge et al. (2014)	EEG	4	3	CSP and SVM	72.3, 73.2
Wang et al. (2014)	EEG	4	9	ICA and SVM	71.8
Naeem et al. (2006)	EEG	4	8	ICA and CSP	Between 33 and 84
Our work	fNIRS	4	26	Time distributed CNN-LSTM	78.44
Our work	EEG	4	26	Time distributed CNN-LSTM	86.24
Our work	Hybrid EEG-fNIRS	4	26	Time distributed CNN-LSTM	88.41

TABLE 3 Comparison with other 4-class hybrid classification studies for BCI with same dataset.

Authors	Brain acquisition modality	Classes	Subjects	Methods	Performance %
Saadati et al. (2020)	EEG-fNIRS	4	26	DNN	87
Kwon et al. (2020)	EEG-fNIRS	3	26	CSP	77.6
Our work	Hybrid EEG-fNIRS	4	26	Time distributed CNN-LSTM	88.41

such as time and frequency feature analysis, event-related synchronization-desynchronization analysis, common spatial or temporal patterns, and spatial-spectral decomposition. Most of these methods require high computational costs and are less feasible to use for real-time BCI (Janapati et al., 2020). With advances in brain signal acquisition modalities, the demands for better signal processing and feature extraction have also increased. Traditional methods of extracting useful information from multi-channel brain signal acquisition modalities, such as time and frequency analysis, event-related synchronization and desynchronization analysis, and finding common spatial and temporal patterns are computationally expensive and not very feasible for real-time BCI applications. RQA of RP has become popular in recent years for analyzing brain activity because brain signals are both recurrent and dynamic. RQA is an analysis technique used to quantify features of the constructed RP. In the literature, RQA features have been used in EEG signal detection of epilepsy and Alzheimer's disease, coupling, and synchronization in EEG of epileptic discharge. Cortical function during different sleep stages was analyzed using RP features. The RQA analysis showed that unique RPs were extracted for different sleep stages (Parro and Valdo, 2018). Several studies have also used SVM and ANNs to classify extracted RQA features. One study used a four-layer ANN for different EEG channels to predict the onset of seizures using RQA measures (Torse et al., 2019).

Considering the complexity and computational cost of DNNs, researchers have invested tremendous efforts to determine the best architecture for deep learning neural networks, giving rise to the sub-research field known as NAS. However, there is no definite conclusion regarding the optimal neural architecture a priori. The number of neurons, number of filters, number of layers, their combinations, dropout, and max-pooling percentage remain the best hyperparameters. The most viable approach seems to be using intuition and domain knowledge to determine an initial guess for these parameters and then iteratively shortlist to obtain good values. In this study, the NAS design process was as follows: a network with a minimum number of parameters, a single convolutional layer, a single LSTM layer, and one dense layer was created; other hyperparameters were tuned; more layers were added, and the network hyperparameters were tuned with a grid search using the sklearn wrapper. We performed the above grid search with sample data and chose the best-performing network for the fNIRS dataset. Another advantage of using an RP with a DNN is that it incorporates the entire signal and does not require any extra steps, such as feature extraction and feature selection. Moreover, it also minimizes extra pre-processing steps, such as finding temporal or spatial features. The constructed RPs of EEG and fNIRS were fed to the classification network to detect the class of activity (0, 2-back, 3-back, or rest). The classification network used was the time-distributed CNN-LSTM. CNN is best known for feature extraction from multidimensional images. In contrast, the RNN has an excellent pattern recognition ability for input sequences. However, CNN and RNN have stability issues due to either exploding or vanishing gradients. An RNN variant was used to solve this issue by using memory cells and LSTM. The highest classification accuracy for four-class mental workload data for the BCI was achieved using this network.

The study results indicate that using the hybrid modalities for the classification of BCI results in higher accuracy than that of the single modality, along with an increase in the number of commands and a reduction in detection time. Figure 8 shows the comparison of the average accuracies achieved for all BCIs. The results of our study have proven the initial hypothesis that incorporating the entire EEG signal for hybrid EEG-fNIRS BCI instead of downsampling will significantly increase classification accuracy. The classification accuracies achieved with our proposed methodology were the highest compared to other classification methods for four-class EEG-based BCI and other studies on the same dataset for four-class classification for hybrid EEG-fNIRS. A comparison with relevant studies is presented in Tables 2, 3, respectively. Moreover, implementing the TD layers resulted in faster and easier computation. This may prove to be a state-of-the-art algorithm in the present BCI realm. The results show a promising future for the use of RPs in real-time BCI. The proposed classification method can help improve the accuracy of real-time BCI.

The limitations of our work are as follows: first, the proposed methodology is computationally costly, and substantial computational resources are required to train and test deep learning models with large datasets because the size of RP increases exponentially with the data size fed at a time, as does the model complexity. Second, the proposed algorithm has not yet been implemented for real-time BCI, which leaves room for network improvement and optimization. There are many potential applications for RPs in BCIs other than EEG and fNIRS signals, as all biological signals constitute dynamic time series data, and our study has validated the successful implementation of RP for time-series data analysis. In the future, further work can be conducted to explore and experiment with new deep learning methods in computer vision along with RPs, such as transformers and attention learning for active channel selection for real-time BCI. Working in this direction will help researchers mitigate nuances related to deep learning algorithms in BCI.

Conclusion

This paper provides an inimitable time-distributed convolutional neural network and long short-term memory method for the integrated categorization of fNIRS-EEG for hybrid BCI applications. The recorded brain signals are first projected onto a non-linear dimension via RPs and supplied into the CNN to extract critical characteristics without downsampling. Then, LSTM is utilized to learn the chronological properties and time-dependence relation to identify brain activity. The average accuracy levels were 78.44% for fNIRS, 86.24% for EEG, and 88.41% for hybrid EEG-fNIRS BCI when using the suggested model. The findings support the RP-based deep-learning algorithm's suitability for effective BCI applications.

Data availability statement

The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Ethics statement

The original study by Shin et al. (2018) was reviewed and approved. The data set is an open-source meta dataset of Shin et al. (2018). The patients/participants provided their written informed consent to participate in the original study. For the current study, ethical review and approval, and written informed consent, were not required, in accordance with the local legislation and institutional requirements.

Author contributions

NM wrote the first draft manuscript. MK conceived the idea of the manuscript and supervised the process. KK, KJ, and HS collected and analyzed the methodologies reported in the paper. NN, UG, and K-SH participated in the revision of the manuscript. NM, MK, UG, and K-SH revised and edited the manuscript. All authors contributed to the article and approved the submitted version.

Funding

This work was supported by the Technology Innovation Program (Grant No. 20016846, Development of fNIRS-based multimonitoring healthcare device for sleep state diagnosis) funded by the Ministry of Trade, Industry, and Energy (MOTIE, Korea).

Conflict of interest

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

Publisher's note

References

Acharya, U. R., Hagiwara, Y., Deshpande, S. N., Suren, S., Koh, J. E., Oh, S. L., et al. (2019). Characterization of focal EEG signals: a review. *Fut. Gen. Comput. Syst.* 91, 290–299. doi: 10.1016/j.future.2018.08.044

Ahn, S., and Jun, S. C. (2017). Multi-modal integration of EEG-fNIRS for brain-computer interfaces—current limitations and future directions. *Front. Hum. Neurosci.* 11, 503. doi: 10.3389/fnhum.2017.00503

Al-Taleb, M. K. H., Purcell, M., Fraser, M., Petric-Gray, N., and Vuckovic, A. (2019). Home used, patient self-managed, brain-computer interface for the management of central neuropathic pain post spinal cord injury: usability study. *J. Neuroeng. Rehabil.* 16, 128. doi: 10.1186/s12984-019-0588-7

Antonietti, A., Balachandran, P., Hossaini, A., Hu, Y., and Valeriani, D. (2021). The BCI Glossary: a first proposal for a community review. *Taylor Francis* 8, 42–53. doi: 10.1080/2326263X.2021.1969789

Asgher, U., Khalil, K., Ayaz, Y., Ahmad, R., and Khan, M. J. (2020a). "Classification of mental workload (MWL) using Support Vector Machines (SVM) and Convolutional Neural Networks (CNN)," in *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies: Idea to Innovation for Building the Knowledge Economy, ICoMET*. pp. 1–6. doi: 10.1109/iCoMET48670.2020.9073799

Asgher, U., Khalil, K., Khan, M. J., Ahmad, R., Butt, S. I., Ayaz, Y., et al. (2020b). Enhanced accuracy for multiclass mental workload detection using long short-term memory for brain–computer interface. *Front. Neurosci.* 14, 584. doi: 10.3389/fnins.2020.00584

Bai, Z., Fong, K. N. K., Zhang, J. J., Chan, J., and Ting, K. H. (2020). Immediate and long-term effects of BCI-based rehabilitation of the upper extremity after stroke: a systematic review and meta-analysis. *J. Neuroeng. Rehabil.* 17, 57. doi: 10.1186/s12984-020-00686-2

Benaroch, C., Sadatnejad, K., Roc, A., Appriou, A., Monseigne, T., Pramij, S., et al. (2021). Long-term BCI training of a tetraplegic user: adaptive riemannian classifiers and user training. *Front. Hum. Neurosci.* 15, 635653. doi: 10.3389/fnhum.2021.635653

Burwell, S., Sample, M., and Racine, E. (2017). Ethical aspects of brain computer interfaces: a scoping review. *BMC Med. Ethics* 18, 60. doi: 10.1186/s12910-017-0220-y

Choi, I., Rhiu, I., Lee, Y., Yun, M. H., and Nam, C. S. (2017). A systematic review of hybrid brain-computer interfaces: taxonomy and usability perspectives. *PLoS ONE* 12, e0176674. doi: 10.1371/journal.pone.0176674

Dehghani, M., Mobaien, A., and Boostani, R. (2021). A deep neural network-based transfer learning to enhance the performance and learning speed of BCI systems. *Brain Comput. Interfaces* 8, 14–25. doi: 10.1080/2326263X.2021.1943955

Fu, Y., Zhao, J., Dong, Y., and Wang, X. (2020). Dry electrodes for human bioelectrical signal monitoring. *Sensors* 20, 3651. doi: 10.3390/s20133651

Gao, X., Yan, X., Gao, P., Gao, X., and Zhang, S. (2020). Automatic detection of epileptic seizure based on approximate entropy, recurrence quantification analysis and convolutional neural networks. *Artif. Intell. Med.* 102, 101711. doi: 10.1016/j.artmed.2019.101711

Gaur, P., Gupta, H., Chowdhury, A., McCreadie, K., Pachori, R. B., and Wang, H. (2021). A sliding window common spatial pattern for enhancing motor imagery classification in EEG-BCI. *IEEE Trans. Instrum. Meas.* 70, 1–9. doi: 10.1109/TIM.2021.3051996

Ge, S., Wang, R., and Yu, D. (2014). Classification of four-class motor imagery employing single-channel electroencephalography. *PLoS ONE* 9, e98019. doi: 10.1371/journal.pone.0098019

Ghafoor, U., Yang, D., and Hong, K.-S. (2022). Neuromodulatory effects of HD-tACS/tDCS on the prefrontal cortex: a resting-state fNIRS-EEG study. *IEEE J. Biomed. Health Inform.* 26, 2192–2203. doi: 10.1109/JBHI.2021.3127080

Hasan, M. A., Khan, M. U., and Mishra, D. (2020). A computationally efficient method for hybrid EEG-fNIRS BCI based on the pearson correlation. *Biomed Res. Int.* 2020, 1838140. doi: 10.1155/2020/1838140

Hong, K.-S., Ghafoor, U., and Khan, M. J. (2020). Brain–machine interfaces using functional near-infrared spectroscopy: a review. *Artif. Life Rob.* 25, 204–218. doi: 10.1007/s10015-020-00592-9

Hong, K.-S., and Khan, M. J. (2017). Hybrid brain-computer interface techniques for improved classification accuracy and increased number of commands: a review. *Front. Neurorobot.* 11, 35. doi: 10.3389/fnbot.2017.00035

Hong, K.-S., Khan, M. J., and Hong, M. J. (2018). Feature extraction and classification methods for hybrid fNIRS-EEG brain-computer interfaces. *Front. Hum. Neurosci.* 12, 246. doi: 10.3389/fnhum.2018.00246

Hong, K.-S., and Zafar, A. (2018). Existence of initial dip for BCI: an illusion or reality. *Front. Neurorobot.* 12, 69. doi: 10.3389/fnbot.2018.00069

Houshyarifar, V., and Amirani, M. C. (2017). Early detection of sudden cardiac death using Poincaré plots and recurrence plot-based features from HRV Signals. *Turk. J. Electr. Eng. Comput. Sci.* 25, 1541–1553. doi: 10.3906/elk-1509-149

Huve, G., Takahashi, K., and Hashimoto, M. (2018). "Brain-computer interface using deep neural network and its application to mobile robot control," in *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC),* 169–174. doi: 10.1109/AMC.2019.8371082

Huve, G., Takahashi, K., and Hashimoto, M. (2019). "Online recognition of the mental states of drivers with an fNIRS-based brain-computer interface using deep neural network," in *2019 IEEE International Conference on Mechatronics (ICM),* 238–242. doi: 10.1109/ICMECH.2019.8722936

Islam, M. K., Ghorbanzadeh, P., and Rastegarnia, A. (2021). Probability mapping based artifact detection and removal from single-channel EEG signals for brain–computer interface applications. *J. Neurosci. Methods* 360, 109249. doi: 10.1016/j.jneumeth.2021.109249

Janapati, R., Dalal, V., Govardhan, N., and Gupta, R. S. (2020). Review on EEG-BCI classification techniques advancements. *IOP Conf. Ser. Mater. Sci. Eng.* 981, 032019. doi: 10.1088/1757-899X/981/3/032019

Jiang, X., Bian, G., and Tian, Z. A. (2019). Removal of artifacts from EEG signals: a review. *Sensors*. 19, 987. doi: 10.3390/s19050987

Kamran, M. A., Naeem Mannan, M. M., and Jeong, M. Y. (2018). Initial-dip existence and estimation in relation to DPF and data drift. *Front. Neuroinform.* 12, 96. doi: 10.3389/fninf.2018.00096

Khan, M. J., and Hong, K.-S. (2017). Hybrid EEG-FNIRS-based eight-command decoding for BCI: application to quadcopter control. *Front. Neurorobot.* 11, 6. doi: 10.3389/fnbot.2017.00006

Khan, M. U., and Hasan, M. A. (2020). Hybrid EEG-fNIRS BCI fusion using multi-resolution singular value decomposition (MSVD). *Front. Hum. Neurosci.* 14, 599802. doi: 10.3389/fnhum.2020.599802

Kim, G. H., Pham, P. T., Ngo, Q. H., and Nguyen, Q. C. (2021). Neural network-based robust anti-sway control of an industrial crane subjected to hoisting dynamics and uncertain hydrodynamic forces. *Int. J. Control Autom. Syst.* 19, 1953–1961. doi: 10.1007/s12555-020-0333-9

Kwon, J., Shin, J., and Im, C. H. (2020). Toward a compact hybrid brain-computer interface (BCI): performance evaluation of multi-class hybrid EEG-fNIRS BCIs with limited number of channels. *PLoS ONE* 15, e0230491. doi: 10.1371/journal.pone.0230491

Ledesma-Ramirez, C. I., Bojorges-Valdez, E., Yanez-Suarez, O., and Pina-Ramirez, O. (2020). "Recurrence analysis in the detection of continuous task episodes for asynchronous BCI," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC),* 1511–1517. doi: 10.1109/SMC42975.2020.9282907

Li, J., Wang, J., Wang, S., Qi, W., Zhang, L., Hu, Y., et al. (2021). Neural approximation-based model predictive tracking control of non-holonomic wheel-legged robots. *Int. J. Control Autom. Syst.* 19, 372–381. doi: 10.1007/s12555-019-0927-2

Liu, Z., Shore, J., Wang, M., Yuan, F., Buss, A., and Zhao, X. (2021). A systematic review on hybrid EEG/fNIRS in brain-computer interface. *Biomed. Signal Process. Control* 68, 102595. doi: 10.1016/j.bspc.2021.102595

Mane, R., Chouhan, T., and Guan, C. (2020). BCI for stroke rehabilitation: motor and beyond. *J. Neural Eng.* 17, 041001. doi: 10.1088/1741-2552/aba162

Mercado, J., Escobedo, L., and Tentori, M. (2021). A BCI video game using neurofeedback improves the attention of children with autism. *J. Multimodal User Interfaces* 15, 273–281. doi: 10.1007/s12193-020-00339-7

Mughal, N. E., Khalil, K., and Khan, M. J. (2021). "fNIRS based multi-class mental workload classification using recurrence plots and CNN-LSTM," in *2021 International Conference on Artificial Intelligence and Mechatronics Systems* (*AIMS*), 1–6. doi: 10.1109/AIMS52415.2021.9466084

Naeem, M., Brunner, C., Leeb, R., Graimann, B., and Pfurtscheller, G. (2006). Seperability of four-class motor imagery data using independent components analysis. *J. Neural Eng.* 3, 208–216. doi: 10.1088/1741-2560/3/3/003

Nayak, S. K., Bit, A., Dey, A., Mohapatra, B., and Pal, K. (2018). A review on the nonlinear dynamical system analysis of Electrocardiogram signal. *J. Healthc. Eng.* 2018, 6920420. doi: 10.1155/2018/6920420

Oh, D. C., and Jo, Y. U. (2021). Classification of hand gestures based on multi-channel EMG by scale Average wavelet transform and convolutional neural network. *Int. J. Control Autom. Syst.* 19, 1443–1450. doi: 10.1007/s12555-019-0802-1

Ortega, P., Zhao, T., and Faisal, A. A. (2020). HYGRIP: full-stack characterization of neurobehavioral signals (fNIRS, EEG, EMG, force, and breathing) during a bimanual grip force control task. *Front. Neurosci.* 14, 919. doi: 10.3389/fnins.2020.00919

Park, J., and Jung, D. J. (2021). Deep convolutional neural network architectures for tonal frequency identification in a lofargram. *Int. J. Control Autom. Syst.* 19, 1103–1112. doi: 10.1007/s12555-019-1014-4

Parro, V. C., and Valdo, L. (2018). Sleep-wake detection using recurrence quantification analysis. *Chaos* 28, 085706. doi: 10.1063/1.5024692

Pham, M., Goering, S., Sample, M., Huggins, J. E., and Klein, E. (2018). Asilomar survey: researcher perspectives on ethical principles and guidelines for BCI research. *Brain Comput. Interfaces* 5, 97–111. doi: 10.1080/2326263X.2018.1530010

Phanikrishna, B. V., Pławiak, P., and Prakash, A. J. (2021). *A Brief Review on EEG Signal Pre-Processing Techniques for Real-Time Brain-Computer Interface Applications*.

Puce, A., and Hämäläinen, M. (2017). A review of issues related to data acquisition and analysis in EEG/meg studies. *Brain Sci.* 7, 58. doi: 10.3390/brainsci7060058

Rahman, M. A., Siddik, A. B., Ghosh, T. K., Khanam, F., and Ahmad, M. (2020). A narrative review on clinical applications of fNIRS. *J. Digit. Imaging* 33, 1167–1184. doi: 10.1007/s10278-020-00387-1

Rashid, M., Sulaiman, N., P. P., Abdul Majeed, A., Musa, R. M., Ahmad, A. F., et al. (2020). Current status, challenges, and possible solutions of EEG-based brain-computer interface: a comprehensive review. *Front. Neurorobot.* 14, 25. doi: 10.3389/fnbot.2020.00025

Rejer, I., and Cieszyński, Ł. (2019). Independent component analysis for a low-channel SSVEP-BCI. *Pattern Anal. Appl.* 22, 47–62. doi: 10.1007/s10044-018-0758-4

Saadati, M., Nelson, J., and Ayaz, H. (2020). Convolutional neural network for hybrid fNIRS-EEG mental workload classification. *Adv. Intell. Syst. Comput.* 953, 221–232. doi: 10.1007/978-3-030-20473-0_22

Saeed, A., Naseer, N., and Jabbar, H. (2020). "Improving classification performance of hybrid EEG-fNIRS BCI system by channel optimization," in *PETRA '20: Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments,* 4, 1–4. doi: 10.1145/3389189.3393747

Sattar, N. Y., Kausar, Z., Usama, S. A., Farooq, U., and Khan, U. S. (2021). EMG based control of transhumeral prosthesis using machine learning algorithms. *Int. J. Control Autom. Syst.* 19, 3522–3532. doi: 10.1007/s12555-019-1058-5

Sazgar, M., and Young, M. G. (2019). *Overview of EEG, Electrode Placement, and Montages*. Cham : Springer 117–125. doi: 10.1007/978-3-030-03511-2_5

Shin, J., von Lühmann, A., Kim, D.-W., Mehnert, J., Hwang, H.-J., and Müller, K.-R. (2018). Simultaneous acquisition of EEG and NIRS during cognitive tasks for an open access dataset. *Sci. Data* 5, 180003. doi: 10.1038/sdata.2018.3

Shoeibi, A., Ghassemi, N., Khodatars, M., Moridian, P., Alizadehsani, R., Zare, A., et al. (2022). Detection of epileptic seizures on EEG signals using ANFIS classifier, autoencoders and fuzzy entropies. *Biomed. Signal Process. Control* 73, 103417. doi: 10.1016/j.bspc.2021.103417

Shoeibi, A., Sadeghi, D., Moridian, P., Ghassemi, N., Heras, J., Alizadehsani, R., et al. (2021). Automatic diagnosis of schizophrenia in EEG signals using CNN-LSTM models. *Front. Neuroinform.* 15, 777977. doi: 10.3389/fninf.2021.777977

Singh, S. A., Meitei, T. G., Devi, N. D., and Majumder, S. (2021). A deep neural network approach for P300 detection-based BCI using single-channel EEG scalogram images. *Phys. Eng. Sci. Med.* 44, 1221–1230. doi: 10.1007/s13246-021-01057-4

Sun, J., Wang, J., Yang, P., Zhang, Y., and Chen, L. (2021). Adaptive finite time control for wearable exoskeletons based on ultra-local model and radial basis function neural network. *Int. J. Control Autom. Syst.* 19, 889–899. doi: 10.1007/s12555-019-0975-7

Takahashi, K., Yokono, R., Chu, C., Huve, G., and Hashimoto, M. (2021). "fNIRS–Based BCI using deep neural network with an application to deduce the driving mode based on the driver's mental state," in *Proceedings of the International Neural Networks Society* (Cham: Springer), 213–219. doi: 10.1007/978-3-030-80568-5_18

Tanveer, M. A., Khan, M. J., Qureshi, M. J., Naseer, N., and Hong, K.-S. (2019). Enhanced drowsiness detection using Deep learning: an fNIRS study. *IEEE Access* 7, 137920–137929. doi: 10.1109/ACCESS.2019.2942838

Thyagachandran, A., Kumar, M., Sur, M., Aghoram, R., and Murthy, H. (2020). "Seizure detection using time delay neural networks and lstms," in *2020 IEEE Signal Processing in Medicine and Biology Symposium* (*SPMB*), 1–5. doi: 10.1109/SPMB50085.2020.9353636

Torse, D. A., Khanai, R., and Desai, V. V. (2019). "Classification of epileptic seizures using recurrence plots and machine learning techniques," in *2019 International Conference on Communication and Signal Processing (ICCSP)* 0611–0615. doi: 10.1109/ICCSP.2019.8697989

Vourvopoulos, A., Jorge, C., Abreu, R., Figueiredo, P., Fernandes, J. C., and Bermúdez i Badia, S. (2019). Efficacy and brain imaging correlates of an immersive motor imagery BCI-driven VR system for upper limb motor rehabilitation: a clinical case report. *Front. Hum. Neurosci.* 13, 244. doi: 10.3389/fnhum.2019.00244

Wang, X.-W., Nie, D., and Lu, B.-L. (2014). Emotional state classification from EEG data using machine learning approach. *Neurocomputing* 129, 94–106. doi: 10.1016/j.neucom.2013.06.046

Wankhade, M. M., and Chorage, S. S. (2021). "Eye-blink artifact detection and removal approaches for BCI using EEG," in *2021 International Conference on Recent Trends on Electronics, Information, Communication and Technology (RTEICT)* 718–721. doi: 10.1109/RTEICT52294.2021.9574024

Yaqub, M. A., Hong, K.-S., Zafar, A., and Kim, C.-S. (2021). Control of transcranial direct current stimulation duration by assessing functional connectivity of near-infrared spectroscopy signals. *Int. J. Neural Syst.* 32, 2150050. doi: 10.1142/S0129065721500507

Zhao, X., Ge, C., Ji, F., and Liu, Y. (2021). Monte Carlo method and quantile regression for uncertainty analysis of wind power forecasting based on Chaos-LS-SVM. *Int. J. Control Autom. Syst.* 19, 3731–3740. doi: 10.1007/s12555-020-0529-z