This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2017.2712712, IEEE Transactions on Cognitive and Developmental Systems

Towards Brain-inspired Learning with the Neuromorphic Snake-like Robot and the Neurorobotic Platform

Guang Chen[1], Zhenshan Bing[1], Florian Röhrbein[1], Jörg Conradt[2], Kai Huang[3], Long Cheng[1], Zhuangyi Jiang[1], and Alois Knoll[1]

***Abstract*—Neurorobotic mimics the structural and functional principles of living creature systems. Modeling a single system by robotic hardware and software has existed for decades. However, an integrated toolset studying the interaction of all systems has not been demonstrated yet. We present a hybrid neuromorphic computing paradigm to bridge this gap by combining the Neurorobotics Platform (NRP) with the neuromorphic snake-like robot (NeuroSnake). This paradigm encompasses the virtual models, neuromorphic sensing and computing capabilities, and physical bio-inspired bodies, with which an experimenter can design and execute both in-silico and in-vivo robotic experimentation easily. The NRP is a public web-based platform for easily testing brain models with virtual bodies and environments. The NeuroSnake is a bio-inspired robot equipped with a silico-retina sensor and neuromorphic computer for power-efficiency applications. We illustrate the efficiencies of our paradigm with an easy designing of a visual pursuit experiment in the NRP. We study two automatic behavior learning tasks which are further integrated into a complex task of semi-autonomous pole climbing. The result shows that robots could build new learning rules in a less explicit manner inspired by living creatures. Our method gives an alternative way to efficiently develop complex behavior control of the ro As SNN is a bio-inspired neural network and the NeuroSnake robot is equipped with a spike-based silicon retina camera, the control system can be easily implemented via spiking neurons simulated on neuromorphic hardware such as SpiNNaker.bot.**

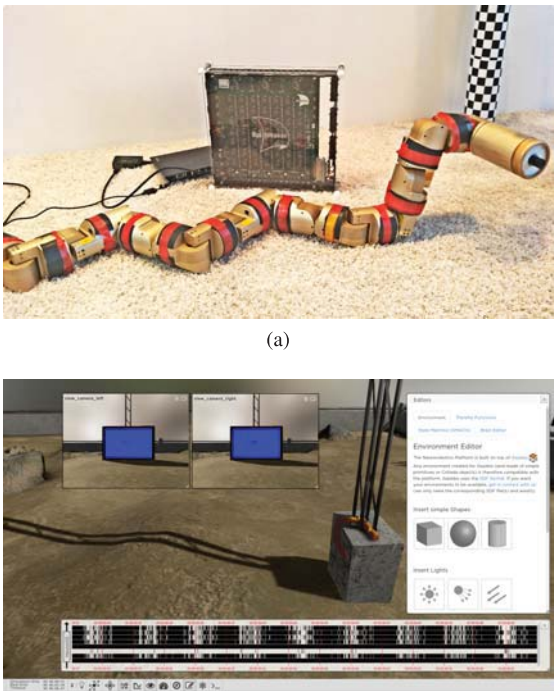
***Index Terms*—Neurorobotics, Neuromorphics, Snake-like Robot, SpiNNaker, Dynamic Vision Sensor**

I. Introduction

THE concept of neurorobotics was firstly introduced in 1933 [1]. A robot was devised with small electromechanical memory cell that could learn its way through a simple maze via conditioned reflex. Since then, however, robotics research has focused mainly on industrial applications [2]. Recently, industry technology has been advancing by developing biologically inspired robots again [2]. With bodies, sensors, and actuators mimicking structural and functional principles at work in the organs of living creatures, scientists studied how the brain, robot body and sensors could work together, enabling a seamless exchange of knowledge between neuroscience and robotics [3].

Authors Affiliation: [1] Technical University of Munich, Faculty of Computer Science, Department of Informatics, [2] Technical University of Munich, Neuroscientific System Theory, [3] Sun Yat-Sen University, School of Data and computer Science. Corresponding author: Zhenshan Bing.

Email: {guang, bing, florian.roehrbein}@in.tum.de, conradt@tum.de, {chengl, jiangz, knoll}@in.tum.de, huangk36@mail.sysu.edu.cn.



(a) (b)

Fig. 1: (a) The *NeuroSnake* robot proof-of-principle setup. The *NeuroSnake* robot is modular designed and equipped with a neuromorphic vision sensor in the head module and a SpiNN-5 48-chip SpiNNaker neuromorphic computer. (b) The virtual *NeuroSnake* robot model is performing the visual pursuit experiment in a virtual environment in the Neurorobotics Platform.

In a typical neurorobotics experiment, a robot or agent will perceive its current environment through a set of sensors that transmit their signals to a simulated brain [2]. Brain models simulating nervous systems that mimic the structure and function of biological brains, are varying levels of details [5]. As the brain is a complex structure, an advanced model is particularly difficult to be simulated in real time. Moreover, neurorobotics studies the interaction among the brain, body, and environment in perception-action closed loops. Thus, modeling brain functions requires the understanding of the interaction of all subsystems in the loop. In the long run, learning complex sensorimotor mapping of the robot generated in the interaction with dynamic and rich sensory environment

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2017.2712712, IEEE Transactions on Cognitive and Developmental Systems

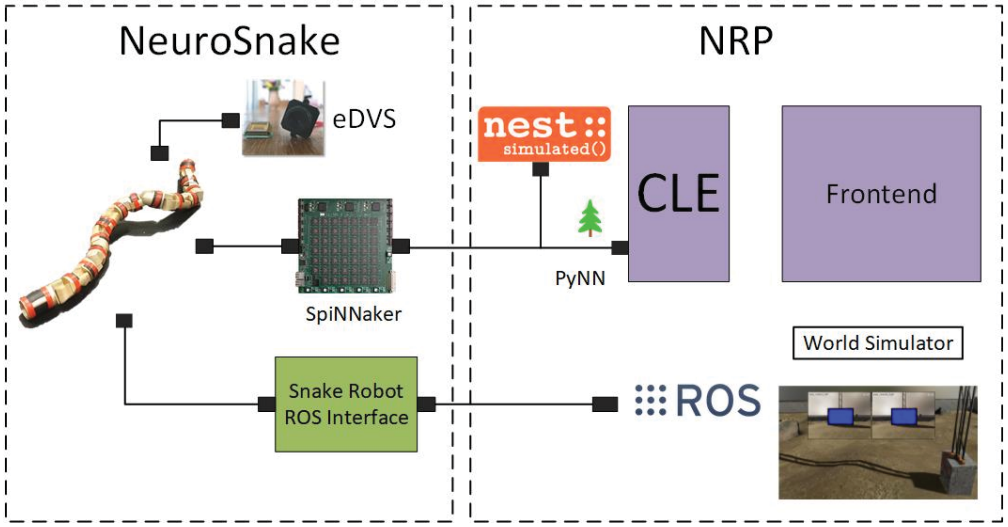


Fig. 2: A hybrid simulation and neuromorphic computing paradigm by combining the Neurorobotics Platform (NRP) [4] and the neuromorphic snake-like robot *the NeuroSnake*. ROS interface connects them to each other. SpiNNaker is able to be integrated into the close loop experiments. For the details of *the NeuroSnake* and NRP, please refer to Section II and Section III respectively.

is also required [6, 7]. For instance, human upper extremity is a highly coupled redundant system. The work of [6] proposed a promising human inspired motion model for the movement controlling, which sheds light on the sensorimotor learning in such complex system. The work is inspired by Qiao's concept [8], which was proposed to achieve high-precision manipulation with low-precision robotic actuators. However, to further foster the understanding of brains in a right way, researchers in neuroscience and robotics face several barriers. These impede the development of modern robots to match the cognitive or behavioral abilities of even the simplest animals, let alone humans.

The first barrier comes from a dilemma for the researchers of neuroscience and robotics: roboticists often use a simplified brain model in a virtual robot to make real-time simulation, or neuro-scientists develop detailed brain models which are not possible to be embedded into a real world due to the high complexity. Although adequate tools to model virtual robots, high-fidelity environments, and complex neural network models already exist, there is so far no public platform that allows neuro-scientist to test neuro-scientific models coupling with virtual robot models [6]. On the other side, robots are still lacking a detailed brain that allows them to exploit the physics of their bodies. As a result, robotic researchers mainly developed machine-learning based on algorithms instead of brain-inspired learning paradigm for pursuing high level intelligence. Although traditional machine-learning approaches took some high level inspirations from biology e.g., structures inspired by the understanding of learning, there is no directly mapping from biological details to robotics.

The second challenge comes from the computing infrastructure of neurorobotics. Neurorobotics applications require strictly real-time execution of large neural simulations [9]. However, the most commonly used computing architecture is still the desktop computer which is under a traditional Von Neuman architecture with many computing cores sharing a common large RAM. Depending on the underlying computing requirement, a conventional computing architecture is not optimized for simulating large neural networks in which all the neurons work in parallel [10]. Although FPGAs and GPUs are more readily available, large-scale neural simulation is still bounded by communication. This issue is well discussed in the recent work [11]. The GPUs and FPGAs are typically better suited for tasks where the ratio of communication/computation favours the latter. In addition, the high power requirements hamper the usability for mobile robots.

To this end, our study looks to demonstrate a hybrid simulation and neuromorphic computing paradigm by combining the Neurorobotics Platform (NRP) [4] and the neuromorphic snake-like robot *the NeuroSnake* (See Fig .2). The NRP allows users to connect pre-defined and customized brain models to detailed simulations of robot bodies and environments in in-silico experiments. The aim of this platform is twofold: from one side, the NRP can be used to test and validate the brain functionalities; on the other side, users can develop a brain-inspired controller by taking advantage of the comprehensive development framework for creating brain, robot and environment models. One pillar of the NRP is that in-silico experiment has the advantage that physics can be slowed down that the robot simulation and detailed brain simulation run in lockstep. The working principles of the brain model in the NRP can be exported to control physical robotics in real world, which address the first barrier mentioned above. Also, by adding a robotic middleware interface such as ROS interface, it is possible to communicate with the real robot through the interface compatible with the NRP.

The *NeuroSnake* is a bio-inspired robot developed in Technische Universität München which is equipped with silicoretina sensors and neuromorphic computing processors, named the SpiNNaker computing platform [12]. The *NeuroSnake* is

a bio-inspired robot consisting of 13 actuated module. The head module has a mininature embedded dynamic vision sensor (*meDVS*) [13] as shown in Fig. 1. The *meDVS* is an address-event silicon retina. Compared to conventional frame-based cameras transmit complete images at fixed latency, the *meDVS* emit events individually and asynchronously at the time they occur, with high temporal resolution, low latency and high dynamic range. The SpiNNaker computing system is a massively-parallel chip multiprocessor system for modelling large systems of spiking neurons in real time. Our previous work demonstrated a scalable approach that could be used to build and control an anthropometric human-scale robot [14]. By taking advantages of these neuromorphic sensor and hardware, the proof-of-principle setup of our neuromorphic snake-like robot offers a prime opportunity to let large neural network interact with and adapt to the real world.

Our main contribution is a hybrid paradigm of simulation and neuromorphic mobile computing platform enabling efficient brain-inspired learning in-silico and in-vivo experiments. We present our NRP platform targeting on the immediate demands of neuro-scientist and robotistics, for the first time to allow the coupling of robots and detailed brain models. We demonstrate the proof-of-principle setup of our neuromorphic snake-like robot which has a unique combination of modular robot design, neuromorphic sensing and computing processors. We illustrate the capabilities and efficiencies of our paradigm with a visual pursuit experiment in the NRP by easily integrating the virtual model of the *NeuroSnake* robot with a simplified brain model, and a semi-autonomous locomotion of the *NeuroSnake* robot with lightweight central pattern generator and automatic behavior learning rules.

II. The Neuromorphic Snake-like Robot: NeuroSnake

The Neurorobotics Platform offers an internet accessible system where scientists from many disciplines can test their brain models. To support both simulated and physical robots and enhance the NRP capabilities, at the Technische Universität München, the Chair of Robotics and Embedded Systems and the Group of Neuroscientific System Theory developed a proof-of-principle setup of a neuromorphic snake-like robot, the *NeuroSnake* as shown in Fig. 2.

A. Modular Snake-like Robot

The NeuroSnake robot is modular designed, consisting of 13 actuated modules, a tail module with power supply, and a special designed head module mounting the *meDVS* [15] sensor as shown in Fig. 1. All the modules are alternately connected with the robot's lateral and dorsal planes. Each module allows a full 180° rotation driven by a servo and a set of gears. The DC servo (DS1509MG) has a maximum torque of $12.8Kg \cdot cm$ and drives a gearbox with a reduction factor of 3.71. For the electronic hardware, each module has an modified arm chip, joint encoder, and other sensors. The arm chip runs three tasks: controlling the servo, reading the joint angle and other sensors information, and communicating with the other modules. The snake robot is power supplied by cables from external power source. Tab. I summarizes the technical specifications of our snake-like robot. The SpiNNaker generates the control signals and transfer them to those modules via I^2C bus. The neuromorphic vision sensor in the head module sends back the real-time event-stream to the host computer via wireless module.

B. Neuromorphic Computing System: SpiNNaker

The SpiNNaker computing system is a massively-parallel chip multiprocessor system for modelling large systems of spiking neurons in real time. It is mainly developed in the University of Manchester [16]. Each SpiNNaker chip has 18 programmable ARM968 cores (200MHz). The communication infrastructure for the processors and inter-chips are specially optimized for massively-parallel computing. It is programmed with the standard neural description languages PyNN [17]. For our NeuroSnake robot, we chose the Nengo open-source neural compiler [18, 19] and the custom Nengo backend that compiles high-level neural models into optimized SpiNNaker code [20].

C. Neuromorphic Vision Sensor: meDVS

The dynamic vision sensor (DVS) [21] is an address-event silicon retina. It detects temporal changes in the perceived illumination and fires pixel location events when a quantized change of log intensity is above a certain threshold [22]. A single event is a tuple (x,y,t,p), where x, y are the pixel coordinates of the event in 2D space, t is the time-stamp of the event, and $p = 1$ is the polarity of the event, which is the sign of the brightness change (increasing or decreasing).

The sensor system used in our *NeuroSnake* Platform is the miniature embedded dynamic vision sensor (*meDVS*), shown in Fig. 1. The *meDVS* is a 128×128 pixel neuromorphic camera. It is preliminary designed for application scenarios where space and weight constraints are crucial, such as our *NeuroSnake* robot. Since the snake-like robot achieves locomotion by twisting its body, the head module is high-frequency rotating and unstable, which can not gather stable visual images by using conventional image sensors. By taking the advantage of high response speed, high dynamic range, and power efficiency, the *meDVS* is a perfect match for the brain-inspired learning tasks such as autonomous locomotion of the *NeuroSnake* robot.

TABLE I: Overview of Snake-like robot specifications

Items	Discriptions
Dimensions	Diameter 60mm Length 70cm
Mass	Module 0.3kg Full 2kg
Actuation	Max Torque 12.8kg.cm Max Speed 0.07sec/60°
Power	24V DC
Communication	I^2C Bus
Sensing	Angular Sensor MLX90316KDC IMU BMI055

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2017.2712712, IEEE Transactions on Cognitive and Developmental Systems

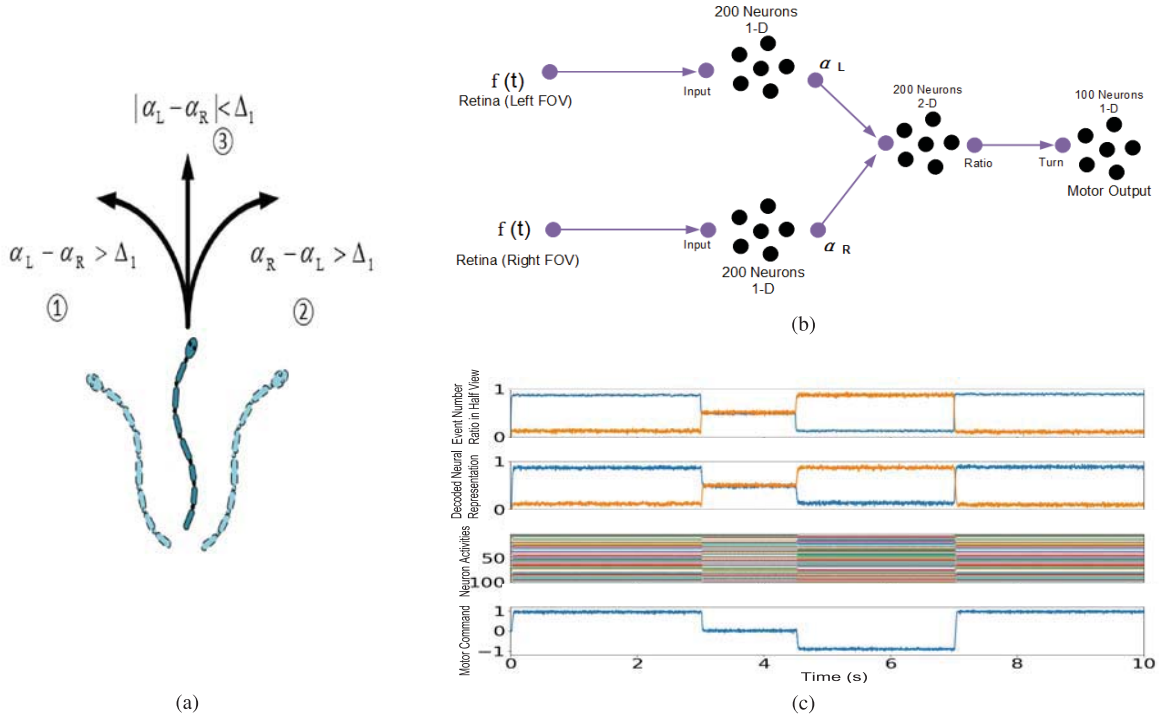


Fig. 3: (a) An example scenario of the *NeuroSnake* robot turning head to show the advantage of the NEF. Two blinking LEDs on each half field of the view of the dynamic vision sensor serve as stimulus. (b) The neural model implemented in the Nengo software includes two *input* populations to encode the processed retina data, one *Ratio* population for the data transformation, and one *motor* population generates the output command. (c) The learned sensorimotor mapping of the neural model by Nengo and NEF. The event streams are preprocessed to calculate the ratios of event numbers for left view and right view of the dynamic vision sensor (row 1). The decoded neural representation of the inputs indicates the accuracy of the neural model (row 2). The neural activity represents the output motor commands (row 3). The neural model is able to generate an approximation of the desired motor commands (row 4), given the processed retina input and target function.

D. The Neural Engineering Framework

As a general purpose neural compiler, the Neural Engineering Framework (NEF) allows users to define a high-level algorithm that is then compiled down to a neural approximation of that algorithm [23]. The first biologically realistic brain model with 2.5 million neurons, named *Spaun* [24], was built with the NEF and the Nengo software. A special simulator called Nengo-SpiNNaker is supported by Nengo, which enable users to run Nengo models in real-time and interacts with external hardware devices such as our *NeuroSnake* robot.

As an example to show the advantage of the NEF for rapidly building the neural models instead of writing customized application, we demonstrate here a task to turn the *NeuroSnake*'s head according to the salient stimulus. The neural model embodied in the agent is presented in Fig 3b. There are three groups of neurons, each representing different values needed in the task. The retina input is a combination of information from 128×128 silicon neurons representing the retinas of the *NeuroSnake* robot. The input populations estimate the ratio of events, α_L and α_R, in half FOV over the total number of events. Two LEDs, randomly blinking at 300 Hz and thus producing a large number of events, are used as the stimulus for the task. Each *input* population is connected to the 200 neurons in the *Ratio* population to form a distributed representation of α_L and α_R. The concept is illustrated in Fig. 3: if one stimulus is present and another stimulus is off (as the status① and ② in Fig. 3a), then the *ratio* population will drive the motor population and send out the left/right command. If both the stimulus are presented or off (as the status③ in Fig. 3a), the motor population will keep the snake head in the center of the field of view.

The implementation is done in the Nengo software, which using the SpiNNaker computing system as the running backend. The NEF is also used to compute the connection weights from the 200 *Ratio* neurons to the 100 *Motor* neurons by encoding the function:

$$M \leftarrow \begin{cases} 1 & \text{if } (\alpha_L \text{ - } \alpha_R) > \Delta_1 \\ -1 & \text{if } (\alpha_R \text{ - } \alpha_L) > \Delta_1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

If we run this model on the snake robot and make the two LEDs randomly blink in left and right FOV, the neural activity of the 100 neurons will change depending on the state of the LEDs. This neural activity can then be decoded (via the NEF) to show how well the neurons are representing this information. The neural model is effectively a neural computational module, where processed spiking retina inputs are fed into the system, manipulated by a sequence of spiking neurons and their connections, and finally decoded and sent to the motors (see Fig. 3c)).

In the experiment section of this article, we have integrated the neural model constructed by the NEF framework with our CPG based locomotion to achieve semi-autonomous locomotion (See Section V). The neural model was built with the Nengo and running on the SpiNNaker Computing system.

III. The Neurorobotic Platform

Compared to the classic robot control paradigm, neurorobotics are controlled by simulated nervous systems which mimic the biological brain at different levels of details. While existing approaches are able to simulate either biological realistic brain model or bio-inspired robots and their interacting world, there is so far no work to facilitate the easily establishment of an interactive research platform of robotics and neuroscience. We therefore developed the neurorobotics platform (NRP) in the scope of the European Unionfunded Human Brain Project[1]. The NRP facilitated a direct link between robotics and neuroscience, enabling a seamless and efficient exchange of knowledge between these two fields. In the following, we briefly describe the architecture and the key components of the NRP. For a detailed introduction of the NRP, we strongly refer to our report paper [4] and our official website[2].

A. Functional Architecture of the NRP

The Neurorobotics Platform is composed of six key software components (see Fig. 4) which are needed to construct neurorobotics experiments from scratch. It can be seen that the NRP provides a complete framework for coupled simulation of robots and brain models. The Brain Simulator simulates the brain by bio-inspired learning algorithms such as spiking neural network to control the robot in a silico neurorobotics experiment. The World Simulator simulates the robots and their interacting environment. The Brain Interface and Body Integrator (BIBI) builds a communication channel between brain models and robot models. The Closed Loop Engine (CLE) is responsible for the control logic of experiments as well as for the data communication between different components. The Backend receives requests from the frontend for the neurorobotics experiment and distributes them to the corresponding component, mainly via ROS. The Frontend is web-based user interface to neurorobotics experiments. Users are able to design a new experiment or edit existing template experiments.

B. Key Components in the NRP

1) Brain Simulator: In the NRP, the Brain Simulator is implemented with spiking neural network (SNN) [25] to simulate a brain circuit. The current supported brain simulator is NEST [26], which is a simulator for spiking neural network models that focus on the dynamics, size and structure of neural systems rather than on the exact morphology of individual neurons.

[1]https://www.humanbrainproject.eu/

[2]http://neurorobotics.net/

2) World Simulator: In order to couple with the accurate brain simulation to perform a realistic neurorobotic experiment, a realistic simulation of the physic world for both the robot and the environment (in which the robot interacts) is required. The Gazebo is chosen and extended as the world simulator inside the NRP. This dynamic simulation can be computed with different supported software libraries like ODE [27] and Bullet [28]. For communicating with the simulated robot, we use the Robot Operating System (ROS) [29] as a middle-ware. The platform uses the asynchronous event-based communication through ROS topics.

3) Brain Interface and Body Integrator: The Brain Interface and Body Integrator (BIBI) is the crucial component in the NRP because it establishes the connection between the robot and brain simulation. The BIBI is composed of a bunch of Transfer Functions which can be defined by the users. We defined two main types of TFs inside the NRP: the Robot to Neuron TFs (R2NTFs) and the Neuron to Robot TFs (N2RTFs). The R2NTFs translate the robot output signals such as images and joint states into neuron signals such as spikes, electric currents or firing rates. The N2FTFs convert the neural signals from neurons into the motor commands for robot motors. Therefore, the perception-action loop of the interaction of the brain, body and environments is closed by these two types of transfer functions.

4) Closed Loop Engine: The Closed Loop Engine orchestrates the two simulations and the data transfer. It is responsible for the synchronization and data exchanges among the above three components. The CLE guarantees that both simulations run at the same time-step, and the TFs run at the end of the simulation steps. A typical execution of a time-step is: after the physics and neural simulations have completed their execution in parallel, the TFs receive and process data from the simulations and produce an output, which is the input for the execution of the next step. The idea behind the proposed synchronization mechanism is to let both simulations run for a fixed time step, receiving and processing the output of the previous steps and yielding data that will be processed in the future steps by the concurrent simulation [4].

5) Backend: The backend is acting as the bridge to connect the user interface and other core components of the NRP. On the user interface end point, it exposes a web server implementing RESTful APIs. It is the first handler for user request, and forwards processed user requests vis ROS on the other end point. The backend in turn provides lists of actions to the user interface which contains simulation listing, simulation handling, simulation creation, experiment listing and manipulation, and backend diagnostic information.

6) Frontend: The Frontend serves as the user interface to the NRP. For the purpose of being accessed and used easily by a broader user base, it is implemented as a web based application using standard web technologies and supporting cross-platform. The Frontend is composed of an Experiment Simulation Viewer and Editors for designing and editing an experiment. The main features of the ESV is that it embeds a high-fidelity 3D view that allows the user to navigate through the virtual environment with the robot model. The Frontend also provides the user with a complete list of editors, *Envi-*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2017.2712712, IEEE Transactions on Cognitive and Developmental Systems

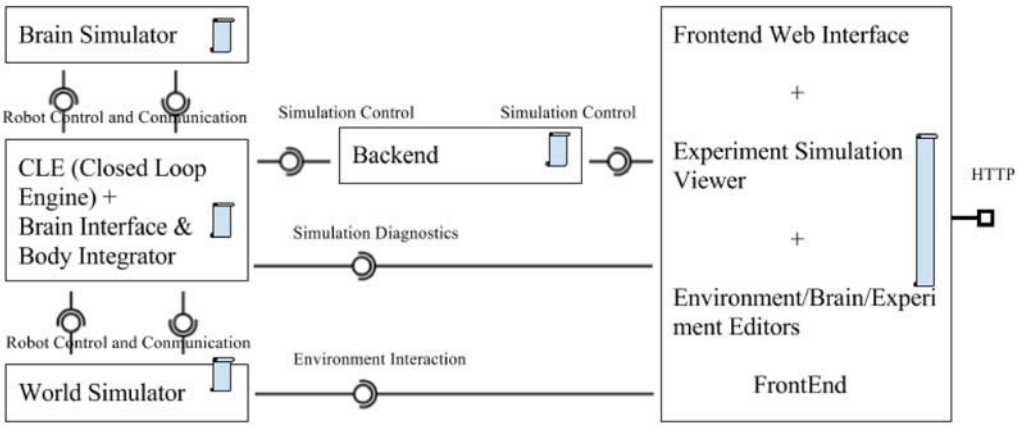


Fig. 4: Functional architecture of the NRP consisting of 6 key software components: Brain Simulator, World Simulator, Brain Interface and Body Integrator, Closed Loop Engine, Backbend, and Frontend.

ronment Editor, Brain Editor, Transfer Function Editor and Experiment Workflow Editor, to configure all the aspects of an experiment.

IV. Autonomous Locomotion of NeuroSnake Robot Based on CPG

Inspired by the morphology of real snakes, the gait of snake-like robots is usually controlled with serpenoid curves [30]. To generate these serpenoid curve signals, one of the most widely adopted control strategy is the Center Pattern Generator. CPG can autonomously produce rhythm signals without external input, which is widely used for snake-like robot locomotion. To achieve autonomous locomotion of the *NeuroSnake* robot, we designed an oscillator model for the chain-coupled CPG network based on the convergence behavior of the gradient system, which can adjust the output signals frequency, phase difference, amplitude and amplitude bias. The flexibility of our CPG network enables the high-level neural models (e.g., neural network learned in Nengo) to control the gaits of the NeuroSnake robot.

A. Central Pattern Generator

For topological structures, CPG networks are roughly classified into two kinds, chain-type and ring-type. The unique chain-type appearance of our *NeuroSnake* robot makes it applicable to use chain-type CPG network. Each module of the robot can be treated as a signal CPG unit, which coupling the neighbour units as a chain as shown in Fig. 5.

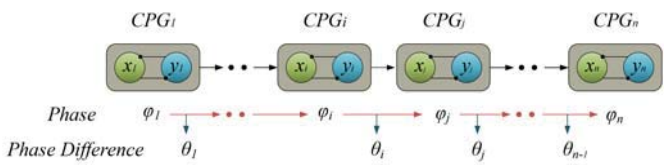


Fig. 5: Parameters setting of CPG net with chain-type.

The mathematic model of one CPG unit consists of two parts, amplitude equations and phase equations. For the first part, two PD controllers are adopted to ensure the convergence of the amplitude [31] and the amplitude bias,

$$\ddot{r}_i = a_i[\frac{a_i}{4}(R_i - r_i) - \dot{r}_i] \tag{2}$$

$$\ddot{c}_i = a_i[\frac{a_i}{4}(C_i - c_i) - \dot{c}_i] \tag{3}$$

where, R_i determines the stable amplitude in finite time. C_i is used to adjust the biased value of the stable amplitude. r_i, c_i are the amplitude and the amplitude bias variables of the i^{th} oscillator, respectively. For the second part, a second-order gradient system is designed to generate the desired signal's phase.

$$\begin{cases} \dot{\varphi}_i &= \omega_i + A\{i,:\} \cdot \Phi + B\{i,:\} \cdot \tilde{\Theta} \\ \ddot{r}_i &= a_i[\frac{a_i}{4}(R_i - r_i) - \dot{r}_i] \\ \ddot{c}_i &= a_i[\frac{a_i}{4}(C_i - c_i) - \dot{c}_i] \end{cases} \tag{4}$$

where $A\{i,:\}$ is the matrix that describes the gradient descent parameters, and $B\{i,:\}$ is a transfer matrix for the desired phase differences. More details and equations are explained in [32]. Φ is the vector of the CPG neurons' phase, and $\tilde{\Theta}$ is the vector of the phase difference among the CPG neurons. The state variable φ_i is the phase of the i^{th} oscillator, respectively. The positive constants a_i and b_i are used to adjust the convergence speed of the amplitude.

$$x_i = c_i + r_i \sin(\varphi_i) \tag{5}$$

The variable x_i is the rhythmic output signal integrated by the phase ϕ_i, the amplitude r_i, and the amplitude bias c_i. Fig. 6 shows the output signals of the proposed CPG model by adjusting different parameters with time.

B. Locomotion Gaits

Snake-like robots can travel with several kinds of gaits, exhibiting diverse motion morphology. Slithering is one of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2017.2712712, IEEE Transactions on Cognitive and Developmental Systems

TABLE II: Description of the parameters in extended gait equation for slithering and rolling gaits

Items	Descriptions	Rolling	Slithering
n	Module subscript	$1 \sim 16$	$1 \sim 16$
N	Module numbers	16	16
C_{odd}, C_{even}	offset in lateral and dorsal plane	0°, 0°	0°, 0°
A_{odd}, A_{even}	Amplitude in lateral and dorsal plane	40°, 40°	60°, 40°
Ω_{odd}	Spatial frequency in lateral plane	0	$n \cdot 3.5/N \cdot \pi$
Ω_{even}	Spatial frequency in dorsal plane	0	$n \cdot 7/N \cdot \pi$
ω_{odd}, ω_{even}	Time frequency in lateral and dorsal plane	π, π	1, 2
x_{odd}, x_{even}	Cycle numbers	0, 0	1.75, 3.5
y, z	Linear coefficient	1, 0	0.3, 0.7

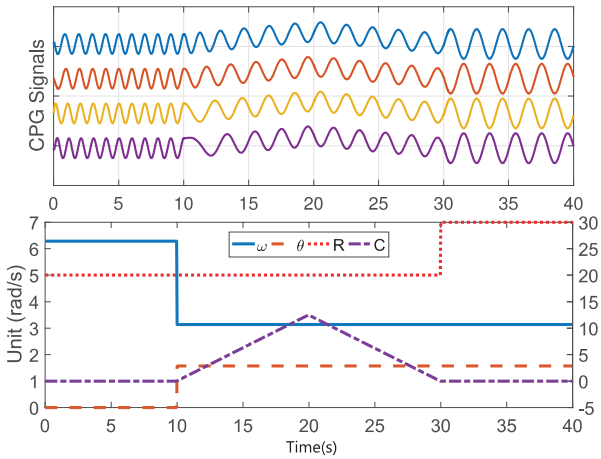


Fig. 6: CPG output waves with changing parameters controlled by high-level brain model. The upper figure shows four CPG output signals. The bottom figure shows different sets of CPG parameters.

the most promising gait for autonomous locomotion, since the robot can travel and observe in the roughly same direction. According to the available literatures, regular gaits can be simply obtained by *gait equation* [33] in 7, such as rolling, sidewinding, and slithering. The meaning of the parameters are depicted in Tab. II.

$$\begin{cases} \alpha(n,t)_{odd} = C_{odd} + A_{odd} \cdot \sin(\Omega_{odd} \cdot n + \omega_{odd} \cdot t) \\ \alpha(n,t)_{even} = C_{even} + A_{even} \cdot \sin(\Omega_{even} \cdot n + \omega_{even} \cdot t + \delta) \end{cases} \quad (6)$$

We adopted a modified slithering gait to achieve autonomous locomotion, because it can ensure the best stability of the meDVS mounted in the head module. Slithering is adopted by snakes to move forward with a S-like shape. The challenge to utilize this gait, is the orientation of the head module, causing by the twist of the whole body. Although the orientation is significantly smaller than other gaits, like rolling or sidewinding, we still have to compensate that instability. Because there is no IMU sensor in the *NeuroSnake* robot at this moment, we have designed a *head scanning* gait to make sure that the the view of meDVS is always horizontal. In a typical autonomous locomotion scenario, this *head scanning* gait is used to identify the target together with the slithering gait. The slithering gait and rolling are modeled as (7). The parameters and values are listed in Tab. II.

$$\begin{aligned} \alpha(n,t)_{even} &= C_{even} + P \cdot A_{even} \cdot \sin(\Omega_{even} \cdot n + \omega_{even} \cdot t) \\ \alpha(n,t)_{odd} &= C_{odd} + P \cdot A_{odd} \cdot \sin(\Omega_{odd} \cdot n + \omega_{odd} \cdot t + \delta) \\ \Omega &= (w + x \cdot \frac{2}{N}) \cdot \pi \\ P &= (\frac{n}{N} \cdot z + y) \quad \in [0,1] \quad \forall n \in [0,N] \end{aligned} \quad (7)$$

V. Experiment

We demonstrate the capabilities and efficiencies of our hybrid simulation and neuromorphic computing paradigm by both in-silico and in-vivo experiments. With the NRP and the virtual model of the *NeuroSnake* robot, we firstly present the designing process of a visual pursuit experiment. The main purpose of this experiment is to show that users can easily connect pre-defined and customized brain models to detailed simulations of robot bodies and envrionments in in-silico experiments. With the *NeuroSnake* robot, we present two automatic behavior learning tasks which are further integrated into a complex task for snake-like robot: *the autonomous pole climbing*. The core concept of these simple automatic learning tasks is to show that our *NeuroSnake* robot is able to build new learning rules in a less explicit manner, inspired by living creatures rather than defining explicit rules from the theory.

A. Visual Pursuit Experiment of NeuroSnake Robot in NRP

The NRP provides a number of design applications for robot, brain and environment models as well as simulation engines. Users can ensemble a *neurorobotics experiment* quickly by constructing a robot model, designing (or selecting) its brain-based controller, and selecting an interactive environment (or uploading customized environment). In principle, researchers can design many kinds of experiments rapidly by matching the specific characteristics of the robot models. In this work, we designed a visual pursuit experiment as a user case to demonstrate the theme of *modular brains for modular bodies* of the NRP platform. We embedded the simplified brains, *Braitenberg vehicles brain* [34], in the appropriate robotic embodiment, *the NeuroSnake robot*. The visual pursuit experiment shows how the virtual *NeuroSnake* robot with a stereo camera head can be controlled by an artificial brain. The virtual *NeuroSnake* robot stands in front of a blue screen with a moving green circle, which is the target of the tracking.

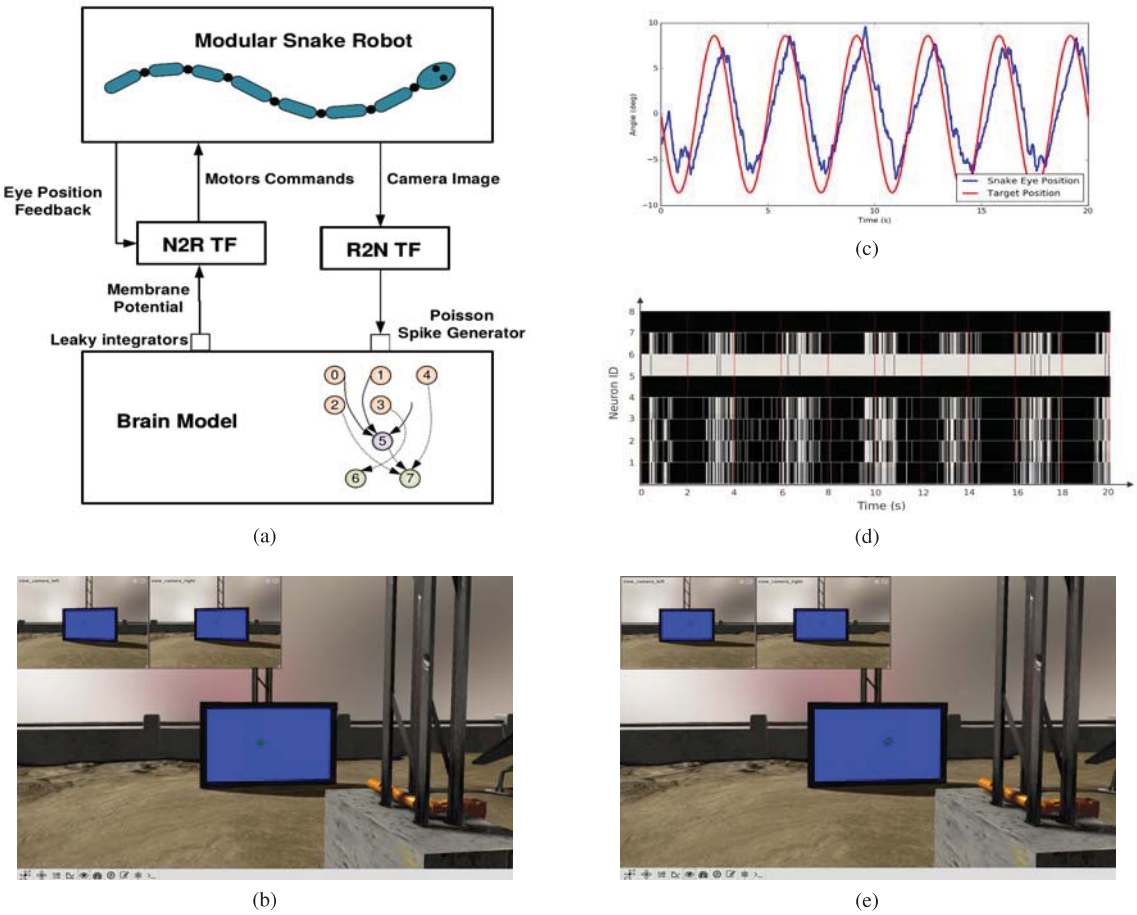


Fig. 7: The *NeuroSnake* visual pursuit experiment. (a) The overall control architecture implemented in the NRP. (b) Snapshot of the *NeuroSnake* performing the visual pursuit inside the NRP when the target moves to the left side. (c) Target and *NeuroSnake* robot eye alongside the x axis during an oscillation frequency of 0.3 Hz. (d) The spikes of the brain network neurons during the experiment (e) Snapshot of the *NeuroSnake* performing the visual pursuit inside the NRP when the target moves to the right side.

The robot reaches and follows such a target using only eye movements.

1) The NeuroSnake Robot Model in the NRP Platform: We created a Gazebo SDF model of the *NeuroSnake* robot with the Blender Robot Designer, which is a standalone application of the NRP platform. The *NeuroSnake* robot model has 13 revolutionary joints for the connections of the snake head, snake body, and snake tail. In the head of the *NeuroSnake* robot model, there are one *eye tilt* joint, two *left eye pan* and *right eye pan* joints, and two virtual joints, named *vergence* and *version*. Two virtual joints compute the difference and the mean between the pans respectively. Each eye also has a camera model with $320 \times 240@60Hz$. Joints are controlled by a gazebo plugin by two PID controllers, one for the position and another for the velocity. The plugin publishes the encoders values to a ROS topic and subscribes to other ROS topics to receive position and velocity commands. The standard Gazebo camera plugin was used for the camera models. In this experiment, we control the eye version, which is directly connected with the *eye pan* joints.

2) The Design of Visual Pursuit Experiment: One advantage of the NRP platform is making the design of neurorobotics experiment as simple as possible. In the rest of this section, we describe the implementation of such an experiment in the NRP with the *NeuroSnake* robot model. Fig. 7a shows the overall control architecture of the *NeuroSnake* visual pursuit experiment. In a visual pursuit task, the target position must be extracted from the camera images, and further be sent to a brain-based controller. Motor commands are then generated and executed to move the gaze towards the targets. To achieve this in the NRP, the target extraction is implemented as Robot to Neuron Transfer functions (R2N TF), while the controller corresponds to a brain model. The command generation is implemented as Neuron to Robot Transfer Functions (N2R TF).

A Robot to Neuron TF translates the image sensor data into the input of the brain model, which comes from the *NeuroSnake* robot model . A simple color filter is applied to extract the target position. In order to feed the object position information to the brain model, the angle of the object α in the camera coordinate is converted with a sigmoidal logistic function: $r = \frac{1}{1+e^{\alpha}}$. The object position is then represented by a parameter r from 0 (closer to the left side) to 1 (closer to the right side), which is further processed by using two

poisson spike generators. Their rates are computed as follows: $r_1 = 1000 \times r$ and $r_2 = 1000 \times (1-r)$. A simplified brain model composed of 8 adaptive exponential integrate and fire neurons was implemented, depicted in Fig. 7a. Among them, neurons from 0 to 4 are sensor neurons, and neurons from 6 to 7 are motor neurons. The neurons labelled as 0 to 3 receive input from the first Poisson spike generator of the Robot to Neuron TF. Neuron 4 receives input from the second spike generator. Neurons 6 and 7 send the output of the Neuron to Robot TF. They are connected to two leaky integrators, implemented with inhibited integrate and fire neurons. The input of the Neuron to Robot TF are the two membrane potentials of these two neurons. The motor command for moving the eyes is computed as a function of the current eye position and a displacement of two membrane potentials such that the eyes move to the opposite directions, when the two motor neurons firing rates differs. The behavior of this brain network is to codify the direction, towards which the eye should move. The experiment is shown in Fig. 7b and Fig. 7e. The *NeuroSnake* robot had to follow a green ball (target) displayed on a blue screen moving with a sinusoidal motion. The performances of the model were tested at various frequencies (from 0.1 to 0.3 Hz), with an amplitude of 18 degrees. It can be observed from Fig. 7c that the model is able to pursue the target the at oscillation frequency of 0.3Hz, by comparing the target and current eye horizontal positions in time.

B. Automatic Learning Experiments with the NeuroSnake Robot

Living creatures observe the world and learn the sensori-motor mapping by random exploration. Similarly, for robotic application, a neural control system should be developed and updated by exploring the environment instead of explicating rules. The core idea is that by exploring the environment in the same way of animals or humans, the robot builds new rules in a less explicit manner by learning the mapping from the recorded sensory to the motor values. This subsection firstly introduced two automatic learning examples with the NeuroSnake robot. Based on the basic behaviors learned in the examples, we demonstrated the autonomous locomotion of our robot in a complex task *the semi-autonomous pole climbing*.

1) Automatic Behavior Learning: It is easy to define an explicit rule for a simple behavior from the theory by a developer e.g., the section IV, however it is infeasible and inefficient for learning complex behaviors without hard-coding rules. We discover the implicit action rule by exploring the environment randomly for the behaviors *automatic turning motion* and *the environment adaptability of slithering gait* of the *NeuroSnake* robot. The implementation is pretty intuitively inspired from the behavior learning principles of humans and animals. Living creatures develop the decision-making rule e.g., *turn left and right* in the real world by randomly performing the right behavior. They remember the sensory data and their responses during these times and increased the likelihood of performing that in the future whenever there is a similar sensory state.

To examine the behavior learning tasks with the *NeuroSnake* robot, we designed two simple behaviors learning cases. The first and simplest learning behavior is *automatic turning motion*, which plays an important role in autonomous locomotion of the snake-like robot. One flashing LED serves as the stimulus in test environment. The *NeuroSnake* achieves the 180-degree scanning by turning the head module, in which the *meDVS* is equipped, to detect the position of the LED. The *NeuroSnake* performs *a head scanning gait* by following *a slithering gait*. The slithering gait randomly turns the snake to right or left by giving an arbitrary amplitude bias defined in Section IV. To eliminate the effect of slithering gait on the relative LED position, the snake robot is moved back manually to the start position while keeping the slithering direction. The *NeuroSnake* calculates the exactly time for turning the head from the start position to the end position where the LED is in the center of view field of the *meDVS*. The *NeuroSnake* records the sensory state (the relative LED position) and the motor value (the amplitude bias) whenever the time taken to perform the head scanning gait is less than the last performing of head scanning gait. With the sensory and motor values, the *NeuroSnake* robot learns the rule for the *automatic turning motion* by using Nengo and NEF. We found that our learning system quickly learn to perform this behavior well, given only 10 examples. The second learning example is *the environment adaptability of slithering gait*. In the second scenario, the robot lied in the center of the two flashing LEDs (each attached to a pole) at a fixed distance in the slithering direction. The robot firstly detects the positions of two LEDs and performs the slithering gait with a random amplitude (see the definition of amplitude in Section IV). We record the sensory state (positions of two LEDs) and the motor values (the amplitude) only when the robot could successfully pass through the two poles. With the same learning paradigm, the robot learns a new rule mapping the recorded sensory states to the amplitude. Having this rule, the robot is able to make the adaptation of slithering gait automatically in a novel situation such as passing through a narrow space.

2) Semi-Autonomous Pole Climbing: For a typical way of autonomous locomotion with snake-like robot, developers are required to design different gaits specifically. However, based on the learning rules in V-B1, we avoid the manually designing processes. We demonstrated the usage of two sets of behaviors *automatic turning motion* and *the environment adaptability of slithering gait* in a complex behavior *semi-autonomous pole climbing*. At the initial position (See Fig. 8a) of a pole climbing scenario, the robot firstly identifies the LED (attached to a pole) by scanning around the environment as shown in Fig. 8b. Based on the detected sensory state (LED position) and the learned rule of the behavior *automatic turning motion*, the robot adopts the left slithering gait automatically to move towards the pole. To prepare the rolling gait for the pole climbing, the robot stops the turning motion at a pre-defined distance and kept its moving direction parallel to the right side of the pole (See Fig. 8c). During the approaching process, the robot stops and scans again to estimate the relative position between the LED and itself (See Fig. 8d). To avoid the future collision with the pole, the robot adjusts the amplitude

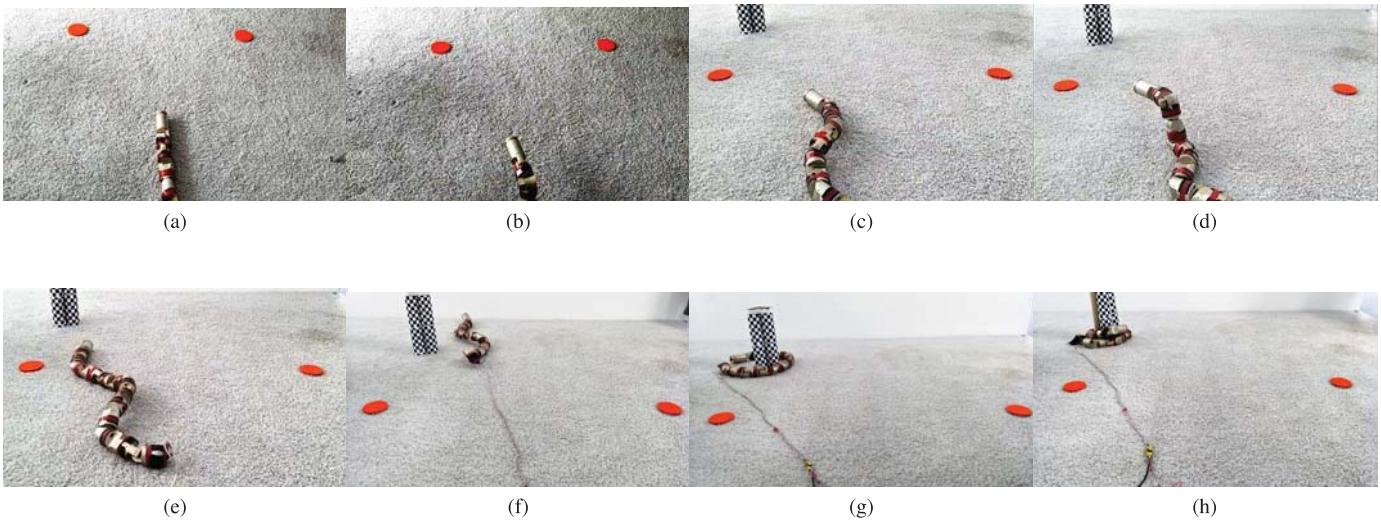


Fig. 8: Montage of the snake-like robot achieves semi-autonomous locomotion. The orange pads are used as the reference positions.

based on the learning rule of the behavior *the environment adaptability of slithering gait* (See Fig. 8e). After getting close to the pole, the snake robot is stopped at a proper position Fig. 8f), where the body is symmetrical spread over the sides of the pole. After that, the last step is conducted, which is to roll sideways to the pole as seen in Fig. 8g). The arc formed by the body will embrace the pole more and more tight. The robot further increases the amplitude of each module to climb up the pole (see Fig. 8h).

3) Results: We have shown that the NeuroSnake robot started from learning simple behavior rules inspired by live creatures, and further performed more complex functions based on these basic learning rules. This approach also makes use of the power-efficiency of neuromorphic sensing and computing processors. Importantly, the neurons learn to approximate these functions which are not defined explicitly. This means that simply giving the robot examples of its desired behavior, robot should be able to develop the learning rules automatically comparing to traditional methods of defining an explicit set of rules. Basic learned rules from simple behaviors gives an alternative way to efficiently develop complex behavior control of the robot.

C. Limitations

It is worth noting that, in the semi-autonomous climbing experiment, we do not claim the learning of our approach is entirely autonomous. In fact, there are human interventions during learning and testing phases of the experiments. The reason is that we are not able to get a high-accuracy 3D position estimation of the LED when the snake-like robot is under slithering gait. The vibration of the snake robot head module makes the DVS sensor unstable. In particularly, the semi-autonomous climbing experiment requires human intervention for two specific tasks: 1. we tell the robot to stop at a pre-defined distance to the pole so that the snake-like robot can adjust the amplitude based on the learning rule of the behavior *the environment adaptability of slithering gait* (See Fig. 8e). 2. We tell the robot to stop at a pre-defined position when the snake is getting close to the pole Fig. 8f). However, this problem is expected to be solved when the snake-like robot equips more sensors in the head module.

VI. Conclusion and Future work

Learning brain-inspired model in an efficient way is not easy. We tackled this problem by proposing a hybrid simulation and neuromorphic computing paradigm. Specifically, we present our NRP platform from a virtual simulation standpoint enabling a seamless and efficient exchange of knowledge between neuroscience and robotics. The NRP is a public platform opening to all users, which for the first time allows the coupling of robots and detailed brain models. We demonstrated the proof-of-principle setup of our neuromorphic snake-like robot and open the opportunity for learning brain-inspired model efficiently on neuromorphic hardware. The *NeuroSnake* has a unique combination of a modular design of snake-like robot, a neuromorphic vision sensor, and a scalable spiking neural network infrastructure (SpiNNaker). We illustrated the capabilities and efficiencies of our paradigm with a visual pursuit experiment in the NRP by easily integrating the virtual model of *NeuroSnake* robot with a simplified brain model. We studied two automatic behavior learning tasks which was further integrated into a complex task for snake-like robot: the autonomous pole climbing. The core concept of these simple automatic learning tasks was to show that the *NeuroSnake* robot was able to build new learning rules in a less explicit manner inspired by living creatures rather than defining explicit rules from the theory. In future, we plan to conduct more realistic experiments with Spiking Neural Network based control system. There are also other possibilities with the NRP and the *NeuroSnake* robot in the future. As the NRP is a ten year project under active development, it is possible to couple the *NeuroSnake* robot to more detailed models of the

brain when the detailed brain is ready. The final goal will be translating virtual robots and brain-derived controllers to physical robotics.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Research and Innovation Programme Horizon 2020 (H2020/2014-2020) under grant agreement NO. 720270 (Human Brain Project).

REFERENCES

[1] T. Ross, "Machines that think," *Health*, vol. 243, p. 248, 1933.
[2] A. Knoll and M.-O. Gewaltig, "Neurorobotics: A strategic pillar of the human brain project," *in Science Robotics*, p. 25, 2016.
[3] H. Qiao, Y. Li, F. Li, X. Xi, and W. Wu, "Biologically inspired model for visual cognition achieving unsupervised episodic and semantic feature learning," *IEEE transactions on cybernetics*, vol. 46, no. 10, pp. 2335–2347, 2016.
[4] E. Falotico, L. Vannucci, A. Ambrosano, U. Albanese, S. Ulbrich, J. C. V. Tieck, G. Hinkel, J. Kaiser, I. Peric, O. Denninger *et al.*, "Connecting artificial brains to robots in a comprehensive simulation framework: the neurorobotics platform," *Frontiers in Neurorobotics*, vol. 11, 2017.
[5] A. K. Seth, O. Sporns, and J. L. Krichmar, "Neurorobotic models in neuroscience and neuroinformatics," 2005.
[6] H. Qiao, C. Li, P. Yin, W. Wu, and Z.-Y. Liu, "Human-inspired motion model of upper-limb with fast response and learning ability–a promising direction for robot system and control," *Assembly Automation*, vol. 36, no. 1, pp. 97–107, 2016.
[7] G. Chen, D. Clarke, M. Giuliani, A. Gaschler, and A. Knoll, "Combining unsupervised learning and discrimination for 3d action recognition," *Signal Processing*, vol. 110, pp. 67 – 81, 2015.
[8] H. Qiao, M. Wang, J. Su, S. Jia, and R. Li, "The concept of attractive region in environment and its application in high-precision tasks with low-precision systems," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 5, pp. 2311–2327, 2015.
[9] H. Qiao, J. Peng, Z.-B. Xu, and B. Zhang, "A reference model approach to stability analysis of neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, no. 6, pp. 925–936, 2003.
[10] M. Migliore, C. Cannia, W. W. Lytton, H. Markram, and M. L. Hines, "Parallel network simulations with neuron," *Journal of computational neuroscience*, vol. 21, no. 2, p. 119, 2006.
[11] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, H. Yang, and W. B. J. Dally, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: ACM, 2017, pp. 75–84. [Online]. Available: http://doi.acm.org/10.1145/3020078.3021745
[12] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, 2013.
[13] F. Galluppi, C. Denk, M. C. Meiner, T. C. Stewart, L. A. Plana, C. Eliasmith, S. Furber, and J. Conradt, "Event-based neural computing on an autonomous mobile platform," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2862–2867.
[14] C. Richter, S. Jentzsch, R. Hostettler, J. A. Garrido, E. Ros, A. Knoll, F. Rohrbein, P. van der Smagt, and J. Conradt, "Musculoskeletal robots: Scalability in neural control," *IEEE Robotics Automation Magazine*, vol. 23, no. 4, pp. 128–137, Dec 2016.
[15] C. Axenie and J. Conradt, "Learning sensory correlations for 3d egomotion estimation," in *Conference on Biomimetic and Biohybrid Systems*. Springer, 2015, pp. 329–338.
[16] S. Furber and S. Temple, *Neural Systems Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 763–796.
[17] F. Galluppi, S. Davies, A. Rast, T. Sharp, L. A. Plana, and S. Furber, "A hierachical configuration system for a massively parallel neural hardware platform," in *Proceedings of the 9th Conference on Computing Frontiers*, ser. CF '12. New York, NY, USA: ACM, 2012, pp. 183–192. [Online]. Available: http://doi.acm.org/10.1145/2212908.2212934
[18] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, p. 48, 2014. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fninf.2013.00048
[19] T. C. Stewart, A. Kleinhans, A. Mundy, and J. Conradt, "Serendipitous offline learning in a neuromorphic robot," *Frontiers in neurorobotics*, vol. 10, 2016.
[20] A. Mundy, J. Knight, T. C. Stewart, and S. Furber, "An efficient spinnaker implementation of the neural engineering framework," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.
[21] J. Conradt, F. Galluppi, and T. C. Stewart, "Trainable sensorimotor mapping in a neuromorphic robot," *Robotics and Autonomous Systems*, vol. 71, pp. 60–68, 2015.
[22] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 $\times$ 128 120 db 15μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb 2008.
[23] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2004.
[24] C. K. Machens, "Building the human brain," *Science*, vol. 338, no. 6111, pp. 1156–1157, 2012.
[25] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International journal of neural systems*, vol. 19, no. 04, pp. 295–308, 2009.
[26] M.-O. Gewaltig and M. Diesmann, "Nest (neural simulation tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.
[27] R. Smith *et al.*, "Open dynamics engine," 2005.
[28] E. Coumans, "Bullet physics engine," *Open Source Software: http://bulletphysics. org*, vol. 1, 2010.
[29] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.
[30] H. Ohno and S. Hirose, "Design of slim slime robot and its gait of locomotion," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 2, 2001, pp. 707–715 vol.2.
[31] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *science*, vol. 315, no. 5817, pp. 1416–1420, 2007.
[32] Z. Bing, L. Cheng, K. Huang, M. Zhou, and A. Knoll, "Smooth gait transition of body shape and locomotion speed based on cpg control for snake-like robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, to appear, June 2017.
[33] M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and scripted gaits for modular snake robots," *Advanced Robotics*, vol. 23, no. 9, pp. 1131–1158, 2009.
[34] D. W. Hogg, F. Martin, and M. Resnick, *Braitenberg creatures*. Citeseer, 1991.

Guang Chen is a senior researcher in fortiss GmbH, a research institute of Technical University of Munich. He was a research scientist in Human Brain Project from 2016 to 2017, a European Commission Future and Emerging Technologies Flagship. He received his Ph.D degree in the Faculty of Informatics in Technical University of Munich, Germany. He received the B.S degree in Mechanical Engineering from Hunan University, China, in 2008, and M.Eng degree in 2013. His research interests range from high-fidelity modeling to functional development of autonomous systems.

Zhenshan Bing is a Ph.D. candidate in the Faculty of Informatics in Technical University of Munich, Germany. He received his B.S degree in Mechanical Design Manufacturing and Automation from Harbin Institute of Technology, China in 2013, and his M.Eng degree in Mechanical Engineering in 2015. His research investigates the snake-like robot which is controlled by artificial neural networks and its related applications.

Florian Röhrbein PD Dr. Florian Röhrbein is a senior lecturer at the research group Robotics and Embedded Systems in TUMs Informatics Department. He is managing director of the Neurorobotics subproject of the Human Brain Project flagship. He has international work experience in various projects on brain-inspired cognitive systems. Research stays include the MacKay Institute of Communication and Neuroscience (UK), the HONDA Research Institute Europe (Germany) and the Albert Einstein College of Medicine (New York). He received his Diploma (with honors) and PhD (magna cum laude) from TU München. In 2011 he received the venia legendi for computer science from Universität Bremen.

Jörg Conradt Prof. Dr. Jörg Conradt is an assistant professor for Neuroscientific System Theory, TU München since 2009. His working group researches information processing in neural systems and the transfer of the principles discovered to technical applications. After studying computer engineering at TU Berlin and USC, Los Angeles, Prof. Conradt completed his doctorate in neuroinformatics/physics at ETH Zurich (2008). He is affiliated with the DFG Cognition for Technical Systems cluster of excellence and the Bernstein Center for Computational Neuroscience. He is the program director ENB Elite Master Program Neuroengineering.

Kai Huang Kai Huang joined Sun Yat-Sen University as a Professor in 2015. He was appointed as the director of the Institute of Cyber Physical Systems of School of Data and Computer Science in 2016. He was a senior researcher in the Computer Science Department, the Technical University of Munich, Germany from 2012 to 2015 and a research group leader in fortiss GmbH in Munich Germany in 2011. He obtained his Ph.D. degree in ETH Zurich, Switzerland in 2010. his MSc from University of Leiden, the Netherlands in 2005, and his BSc from Fudan University, China in 1999. His research interests include techniques for the analysis, design, and optimization of embedded systems, particularly in the automotive domain. He was awarded the Program of Chinese Global Youth Experts 2014 and was granted the Chinese Government Award for Outstanding Self-Financed Students Abroad 2010.

Long Cheng Long Cheng received the B.S degree in Mechanical Design Manufacturing and Automation from Harbin Institute of Technology, China, in 2011, and M.Eng degree in Mechanical Engineering from the same university in 2013. He is currently a Ph.D. candidate in the chair of Robotics and Embedded System at Technical University of Munich. His research interests include bio-inspired robotics, real-time embedded systems, thermal management design.

Zhuangyi Jiang Zhuangyi Jiang is a Ph.D. candidate in the Department of Computer Science in Technical University of Munich, Germany. He received his B.S. degree in Information Security from Fuzhou University, China in 2013, and his M.S. degree in Information Security from Wuhan University, China in 2016. His research investigates the snake-like robot which is controlled by artificial neural networks and robot vision.

Alois Knoll Alois C. Knoll received the diploma (M.Sc.) degree in Electrical/Communications Engineering from the University of Stuttgart, Germany, in 1985 and his Ph.D. (summa cum laude) in Computer Science from the Technical University of Berlin, Germany, in 1988. He served on the faculty of the Computer Science department of TU Berlin until 1993. He joined the University of Bielefeld, as a full professor and the director of the research group Technical Informatics until 2001. Since 2001 he has been a professor at the Department of Informatics, TU München. He was also on the board of directors of the Central Institute of Medical Technology at TUM (IMETUM). From 2004 to 2006, he was Executive Director of the Institute of Computer Science at TUM. Between 2007 and 2009, he was a member of the EUs highest advisory board for information technology, ISTAG, the Information Society Technology Advisory Group, and a member of its subgroup for Future and Emerging Technologies (FET). In this capacity, he was actively involved in developing the concept of the EUs FET Flagship projects. His research interests include cognitive, medical and sensor-based robotics, multi-agent systems, data fusion, adaptive systems, multimedia information retrieval, model-driven development of embedded systems with applications to automotive software and electric transportation, as well as simulation systems for robotics and traffic.