2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

End to End Learning of a Multi-layered SNN Based on R-STDP for a Target Tracking Snake-like Robot

Zhenshan Bing[1], Zhuangyi Jiang[1], Long Cheng[2], Caixia Cai[3,*], Kai Huang[2] and Alois Knoll[1]

Abstract— **This paper introduces an end-to-end learning approach based on Reward-modulated Spike-Timing-Dependent Plasticity (R-STDP) for a multi-layered spiking neural network (SNN). As a case study, a snake-like robot is used as an agent to perform target tracking tasks on the basis of our proposed approach. Since the key of R-STDP is to use rewards to modulate synapse strengthens, we first propose a general way to propagate the reward back through a multi-layered SNN. Upon the proposed approach, we build up an SNN controller that drives a snake-like robot for performing target tracking tasks. We demonstrate the practicability and advantage of our approach in terms of lateral tracking accuracy by comparing it to other state-of-the-art learning algorithms for SNNs based on R-STDP.**

I. Introduction

Recent developments in neuroscience have highlighted that spiking neural networks are much more plausible and realistic for modeling the underlying mechanism of the brain. For living creatures, biological neurons use impulses or spikes to process information and communicate. Since these spikes are typically sparse, it allows feasibly power-efficient and fast-computing implementation of large SNNs [1], [2]. For instance, human brains can carry out visual pattern analysis and classification in just 100 *ms*, in spite of the fact that it involves a minimum of ten synaptic stages from the retina to the temporal lobe [3]. Therefore, SNNs may offer an efficient way not only to model the principles underlying neural structures devoted to locomotion control in living creatures, but more importantly to be used for controlling mobile robots to perform tasks autonomously [4].

Since SNNs use non-differential spikes to communicate, the well-known error back-propagation mechanism is no longer applicable for training SNNs. For the same reason, there has been a void of practical learning rules for mobile robotic applications [5]. Some work constructed their SNNs with manually adjusted weights by mimicing the different function units in the brain [6], [7], [8]. However, these methods have already been burdensome and inefficient for lightweight network with few connections, let alone for solving complex networks.

Recently, the functionality of STDP has revealed that the synaptic connections are affected by the precise timing of pre- and postsynaptic spikes. Meanwhile, STDP learning rule has been used in robotic control [4]. Arena [9] presented an SNN controller based on an unsupervised STDP learning paradigm, which allowed the robot to learn how to navigate in an unknown environment. Bouganis [10] used supervised learning to train a single-layer network to control a robotic arm with four degrees of freedom in 3D space. However, STDP learning rule usually requires a supervisor or past experience, which is infeasible for mobile robotic applications in undiscovered environments.

On the other hand, neuroscience studies show that the brain modifies the outcome of STDP synapses using one or more chemicals emitted by given neurons. This mechanism inspires a new method for training SNNs and is known as reward-modulated spike-timing-dependent plasticity (R-STDP) [11], [12]. Since R-STDP can modulate an SNN with external signals that even sparse or delayed, this method is well suited for mobile robotic tasks. Mahadevuni [13] solved an goal approaching task by training an SNN using R-STDP. Shim [14] further proposed a multiplicative R-STDP by multiplying the current weight to the normal R-STDP and assigned the global award to all the synapses among two separated hidden layers in an SNN.

Even there has been some initial attempts that investigate robotic control applying SNNs based R-STDP, there still exist many problems for widespread implementations. First and foremost, there has been lacking of a unified learning rule that can be easily applied to different tasks and assign neuron modulations regardless the multi-layered SNN structure. Second, the strategies of information encoding and decoding for SNNs are still unclear. Since both typical sensor data and motor command are time-based, information has to be converted into spikes and interpreted from spikes as well.

To this end, our paper looks to propose an applicable multi-layered SNN learning algorithm based on R-STDP learning rule and implement it for end-to-end control in robotics domain. Our main contributions are summarized as follows: First, on the basis of standard R-STDP, we propose a policy that propagates the reward back through all the layers of an SNN and acts as an individual neural modulator. Second, we offer a general workflow to implement an SNN controller based on our proposed learning rule, together with information encoding and decoding strategies. A target tracking task for a snake-like robot is given as a case study. Finally, our SNN-based controller is simulated in different tracking scenarios to demonstrate its adaptability. The results also reveals that compared to other SNN-based controllers with different topologies, our method has advantages in terms of tracking accuracy and stability.

II. Target Tracking Tasks

In order to elaborate our SNN controller clearly, we take target-tracking tasks as case studies for learning and testing

Authors' Affiliation: [1] Department of Informatics, Technical University of Munich, Germany. [2] Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education, China; School of Data and Computer Science, Sun Yat-sen University, China. [3] Institute for Infocomm Research (I2R), A*STAR, Singapore. * Corresponding author.
Email: {bing, jiangz, knoll}@in.tum.de, chenglong3@mail.sysu.edu.cn, huangk36@mail.sysu.edu.cn, cai_caixia@i2r.a-star.edu.sg

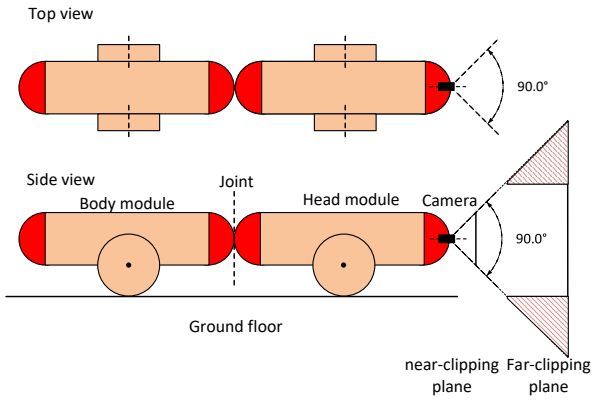


Fig. 1: Views of the first two modules of the snake-like robot.

with a snake-like robot.

The snake-like robot consists of eight joints and nine identical body modules equipped with passive wheels to facilitate lateral undulation. An infrared vision sensor mounted at the front of the head module is used to perceive the information of the target. As we can see from Fig. 1, the perspective angle of the camera is set as 90° and its detecting range is between 0.1 m and 5 m. The shadow areas are cropped for the reason of simplicity. Second, an easily detectable sphere is used as the target that has a radius of 0.15 m and is placed in front of the snake at the initial position. This target is set with a higher temperature and all its surroundings are with the same base temperature. Thus, it can be easily detected by the infrared camera. The target trajectory for training is pre-designed, which is calculated as a sinusoid curve as

$$\begin{cases} x_{target} = t + x_0 \\ y_{target} = s \cdot A_{target} \sin(\pi \times t / \tau_{target}) + y_0 \end{cases} \quad (1)$$

, where x_{target}, y_{target} are the coordinate positions of the target and (x_0, y_0) is the initial coordinate. A_{target} and τ_{target} are constant parameters to modulate the trajectory curve. s is a factor of $+/-1$ that alternates the target's direction each time when the simulation resets, so that the robot is confronted equally with the target going left or right with respect to its vision sensor. In addition, we also design three more challenging tracks for evaluating the SNN controller as unknown scenes, which are a double-frequency sinusoid

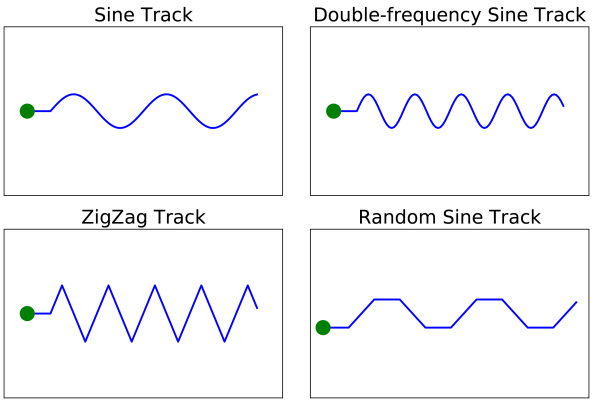


Fig. 2: The pre-designed target trajectories. The first sine track is used as training task. The other three tracks are used as testing tasks.

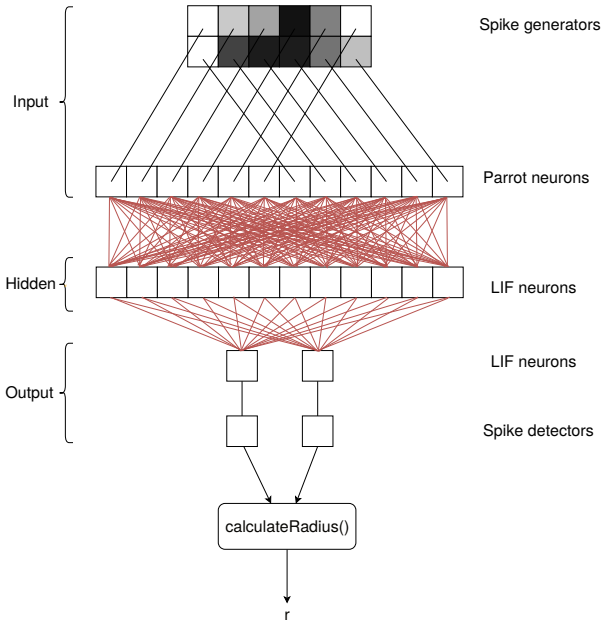


Fig. 3: A graphical abstraction of the structure of the SNN with an agnostic hidden layer.

curve as the training scene, a zigzag curve, and a ladder curve (See Fig. 2). In all these tracks, there is a green dot that indicates the starting position and a short straight line for the beginning process. All the physical simulations are performed in V-REP [15].

III. SNN Controller

In this section, a multi-layered SNN constructed with R-STDP synapses is presented, together with its encoding and decoding strategies. Our SNNs are construed with NEST [16].

A. SNN Architecture

The architecture of the target tracking SNN controller is shown in Fig. 3, which takes infrared image as the input and generates steering commands for the snake-like robot as the output. For the input, the infrared vision sensor data is used to stimulate poisson neurons to generate spike trains. The input layer is connected to a hidden layer with the same size of the input neurons. These neurons in hidden layer are then connected to two leaky-integrate-and-fire (LIF) output neurons for steering the robot by calculating the turning radius. All the connections are using R-STDP synapses in an "all-to-all" fashion.

B. Information Encoding

The incoming heat map image from the infrared camera is translated into the network's input layer as spikes. In order to ease the computation burden, the image data is given to the input neurons as follows. The image with resolution 64 $\times$ 64 pixels is first reduced to 64 $\times$ 16 pixels by cropping the top and bottom by 24 pixels each, since they contain no target information. Then, it is further downsampled to 16×4 pixels by integrating a patch of 4 $\times$ 4 pixels into one input neuron. This yields a single network with $16 \times 4 = 64$ input neurons. The mapping process is visualized in Fig. 4.

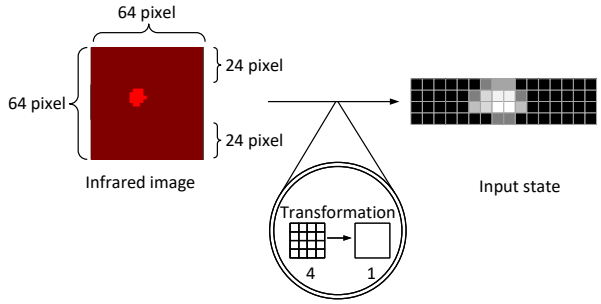


Fig. 4: Visual input encoding strategy. In the right image, a black pixel means a normalized intensity sum of 0 whereas a white pixel corresponds to the maximum intensity sum.

Every pixel has an intensity value in the range of $[0, 255]$ according to their infrared radiation. This value per pixel is then normalized, so that the sum of the intensities of 16 pixels varies between 0 and 16. Again, the resulting value is normalized and then multiplied by the maximum firing rate of 300 Hz. This value is then assigned to the corresponding spike generator for as the mean firing rate.

C. Snake Steering and Information Decoding

For interpreting the neuron activities as motor commands, the output spikes should be decoded to steer the slithering locomotion of a snake-like robot.

Therefore, we first introduce the slithering locomotion of a snake-like robot, which is inspired by the *serpentine curve* [17], [18] and the joint angle ϕ_i is calculated as

$$\phi_i = A \cdot (1 - e^{-\lambda t}) \cdot \sin(\omega\, t + (i-1)\ \beta) + C. \tag{2}$$

A is the joint amplitude and C is its bias to steer the robot. ω is the temporal frequency to propagate the wave through the robot joints and β is the spatial frequency to domain the wave cycles. $1 - e^{-\lambda t}$, as a damping function, is used to smooth the start-up process, where λ is used to adjust the convergence rate. Since the gait modeling is not within the scope of this paper, we just give the relationship between the turning radius R and the bias C, which is only determined by the mechanical properties and gait parameters.

$$R = \frac{l_0 \times \sum_{k=1}^{N} \cos(A \times \sum_{p=1}^{k} \sin(\omega \times p))}{C} \tag{3}$$

, where $N = 8$ is the joints numbers and l_0 is the body module length.

To this point, we can steer the snake-like robot with desired radius by just computing the bias C in (3), which will be added or subtracted for steering the robot left or right. The decoding strategy is inspired by [19], [20] and also adopt a steering wheel model based on an agonist-antagonist muscle system to compute the bias C. First, the output spike count is scaled by the maximum possible output:

$$m_t^{left/right} = \frac{n_t^{left/right}}{n_{max}} \in [0;\ 1], \text{ where } n_{max} = \frac{T_{sim}}{T_{refrac}}. \tag{4}$$

T_{sim} denotes the simulation time step length and T_{refrac} describes the refractory period length of the LIF neuron. Based on the difference of the normalized activities m_t^{left} and m_t^{right} and a turn constant k, the amplitude bias C is calculated as

$$C = k \cdot a_t \text{ , where } a_t = m_t^{left} - m_t^{right} \in [-1;\ 1]. \tag{5}$$

When the snake cannot see the target, the network will receive no input and both outputs will be zero. In this case the network is designed move to the previous turning angle as it remembers, since the target get lost continuously in one side of the vision. To address this issue, the bias for the current step C_t is calculated as

$$C_t = \tau \cdot C + (1 - \tau) \cdot C_{t-1}, \tag{6}$$

$$\text{with } \tau = \sqrt{\frac{(m_t^{left})^2 + (m_t^{right})^2}{2}}. \tag{7}$$

Thus far, we have a decoding method that translate the discrete network's spiking pattern into continuous motor control signals to steer a snake-like robot.

IV. R-STDP Learning

In this section, the R-STDP learning rule is introduced together with the approach for back-propagating the reward for all the R-STDP synapses. Then, the training details based on this learning rule are given as well.

A. Reward-modulated STDP

As the basis of R-STDP, the STDP learning rule models the synaptic weight change in dependence on the time difference between firing times of pre- and post-synaptic neurons and is given as

$$\Delta t = t_{post} - t_{pre} \tag{8}$$

$$W(\Delta t) = \begin{cases} A_+ e^{-\Delta t/\tau_+}, & \text{if } \Delta t \geq 0 \\ -A_- e^{\Delta t/\tau_-}, & \text{if } \Delta t < 0 \end{cases} \tag{9}$$

$$\Delta w = \sum_{t_{pre}} \sum_{t_{post}} W(\Delta t) \tag{10}$$

, where w is the synaptic weight. Δw is the change of the synaptic weight. t_{pre} and t_{post} stand for the timing of the firing spike from pre-neuron and post-neuron. A_+ and A_- represent positive constants scaling the strength of potentiation and depression, respectively. τ_+ and τ_- are positive time constants defining the width of the positive and negative learning window.

In the R-STDP, the synaptic weight w is updated based on the integration of STDP and a reward signal R. First, an eligibility trace of the synapse is used to collect the changes generated by STDP, thus it can work with a delayed reward signal. The eligibility trace E can be defined as

$$\dot{E}(t) = -\frac{E}{\tau_E} + W(\Delta t)\delta(t - s_{pre/post})E_1, \tag{11}$$

where the contribution of all spike pairings with the second spike time at $t - s_{pre/post}$. $s_{pre/post}$ means the time of a pre- or postsynaptic spikes. $E(t)$ decays exponentially with the time constant τ_c. The decay rate controls the sensitivity of plasticity to delayed reward. E_1 is a constant coefficient to modulate the decay rate. W is the synaptic change under STDP mechanism defined in (9).

Algorithm 1 Reward back propagation

The SNN has X layers
The X_i layer has Y_i neurons
$r_1 \leftarrow r$ in (14)
for $i = 1 \rightarrow (X-1)$ **do**
    for $j = 1 \rightarrow Y_{i+1}$ **do**
        $$r_{i+1,j} = \frac{\sum_{k=1}^{Y_i}(r_{i,j,k} \cdot w_{i,j,k})}{\max(w_{i+1,j,1}, \cdots, w_{i+1,j,(Y_i-1)}, w_{i+1,j,Y_i}) \cdot Y_i}$$
    end for
end for

Then, the R-STDP learning rule for updating weights is defined as

$$\dot{w}_{ji}(t) = r(t) \times E_{ji}(t), \tag{12}$$

which indicates that the weight of a synapse is by the product of eligibility trace and reward signal. w_{ji} is the weight of a synapse from neuron i to neuron j, $E_{ji}(t)$ is an eligibility trace of this synapse which collects weight changes proposed by STDP, and $r(t)$ results from a neuromodulatory signal.

Since the reward is directly related to the actions of the robot, we can regard this as an error from the environment to the network. Once those output neurons receive the error, they can back-propagate the error and change the synapse under the effect of R-STDP. For the aforementioned SNN architecture, the synapses connecting the output neurons are directly given the reward r from the environment. However, the reward for the synapses connecting the input and hidden neurons are unknown. In this paper, we propose a method to achieve the "backpropagation" by reversing the dopamine modulation according to different synaptic weights. The pseudo-code for this algorithm is shown in Algorithm 1. Let us assume there is an SNN with X layer including one input layer, one output layer, and $X-2$ hidden layer. The output layer will be regarded as X_1. For the i^{th} layer, there are Y_i neurons. $r_{i,j}$ is the reward given to the j^{th} neuron in the i^{th} layer. $w_{i+1,j,k}$ is the synapse connecting the j^{th} neuron in the $(i+1)^{th}$ layer and the k^{th} neuron in the previous i^{th} layer. The reward given from the environment is regarded as r_1 for each output neuron. In order to calculate the reward given from one neuron, we first sum up all the $r \cdot w$ from those neurons connecting it in the previous layer. Then we normalize it by dividing the maximum synapse value and the neuron amount of those connections. Thus far, we can finally propagate the reward through any multi-layered SNN and an example will be given afterwards.

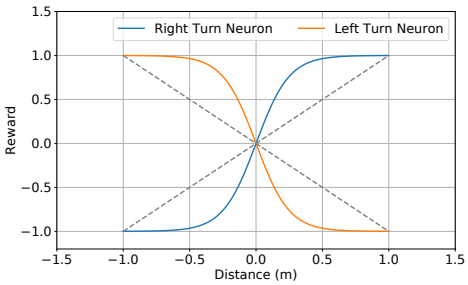


Fig. 5: Reward definition for target the tracking task.

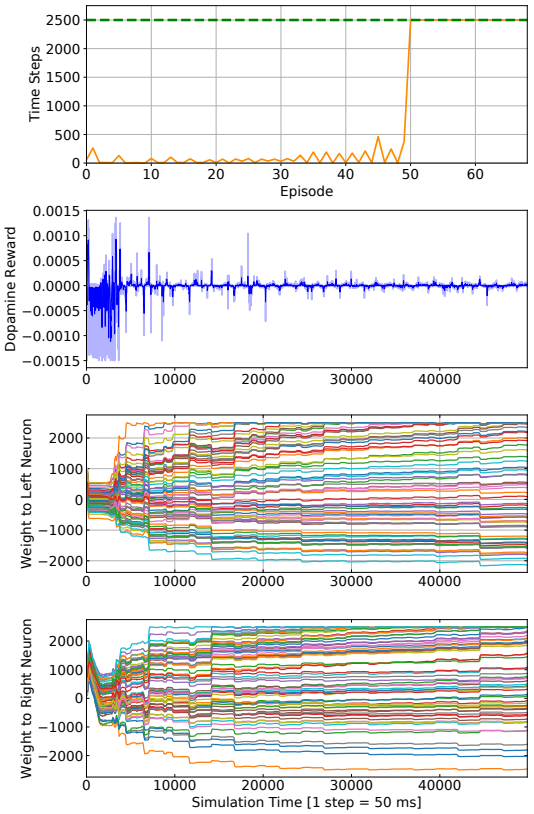


Fig. 6: Training details of the SNN controller are recorded every 100 time steps.

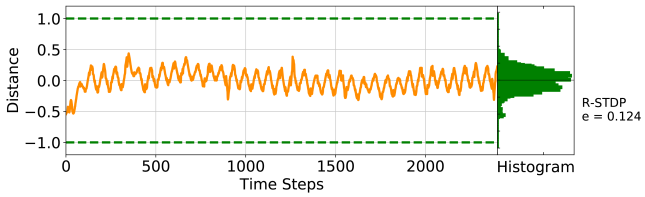


Fig. 7: The target position in the visual field of the robot and its deviation from the visual center.

B. Reward Definition

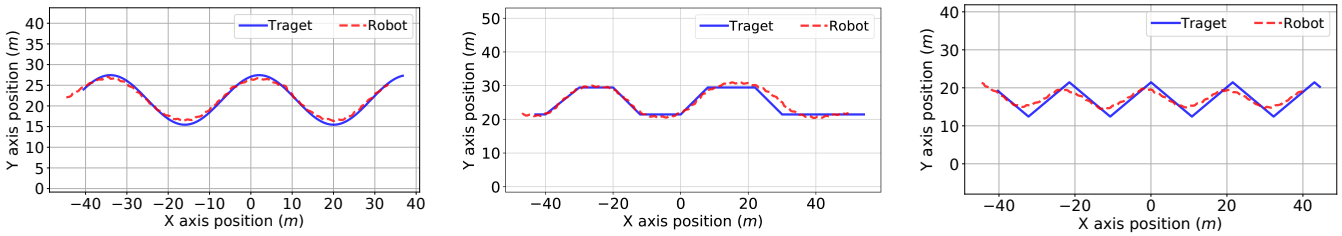
As the synapse plasticity is modeled as R-STDP, a value for this reward-like variable needs to be calculated every simulation step. This value should somehow reflect the good or bad actions taken by the robot.

First, the image from the infrared sensor is processed by only extracting its heat map. Second, the image moments is calculated to compute the centroid of the target d in the visual field, which ranges from $[-1,1]$ and 0 indicates the middle position in the visual field. Thus, the reward r is defined according to the position d as,

$$d = \frac{2 \cdot M_{10}}{M_{00} \cdot 64} - 1, \; d \in [-1,1]. \tag{13}$$

$$r = \frac{2}{1+e^{-d \cdot \gamma}} - 1, \text{ with } \gamma = 8. \tag{14}$$

M_{10} and M_{00} are the first-order and zero-order moments of the image. γ is used to modulate the gradient of the reward



(a) Double-frequency sine track. (b) Ladder shape track. (c) Zigzag track.

Fig. 8: Three testing tracks. In each task, the target track is presented with blue solid line and the robot trajectory is marked with red dash line.

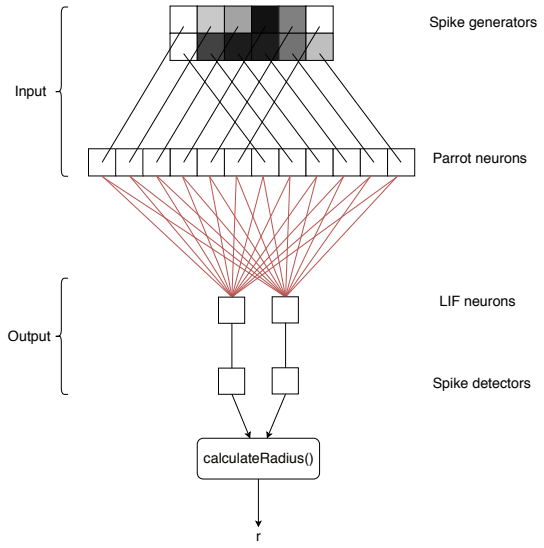


Fig. 9: A graphical abstraction of the structure of the SNN without a hidden layer.

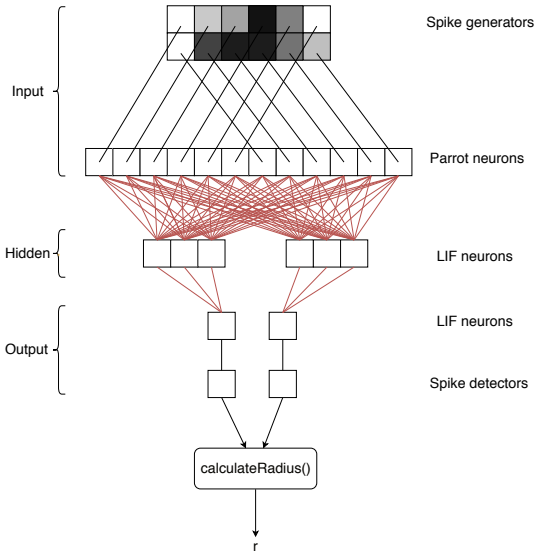


Fig. 10: A graphical abstraction of the structure of the SNN with a separated hidden layer.

curve. From Fig. 5, we can see that the reward r is zero when it is located at the center of the visual field. Thus, the reward modulation stops working and the two steering neuron activities are balanced to each other to drive the snake forward. The reward gets steep when it deviates to the boundaries and turns to its maximum or minimum value when it exceeds ± 0.6. This can help the robot do its best from loosing the target outside of the visual field.

According to Algorithm 1, the reward from the i^{th} neuron in the hidden layer to the input layer can be calculated as

$$r_i = \frac{r \cdot (w_{left} - w_{right})}{max(w_{left}, w_{right}) \cdot 2}, \tag{15}$$

where $w_{left/right}$ are the synapses connecting the left and right output neurons.

C. Training Details

A simulation step is equivalent to 50 *ms* both for the simulation of the SNN and the robot simulator itself. In Fig. 6, the training progress of the R-STDP controller is shown with its steps in each episode, reward, and synaptic weights for left and right output neurons. Specifically, it first shows the time steps in each episode until that the robot loses the target for 20 consecutive time steps or reaches the maximum steps 2500. Correspondingly, the dopamine reward varies irregularly at the beginning and levels off to zero over time steps. In the beginning of the training procedure, the snake-like robot will just slither randomly, since all connection weights for both steering neurons have been randomly set in the range of $100 - 1000$. Therefore, during the first 30 episodes, trials are mostly terminated when the target makes its first turn and drifts out of the vision field of the robot. Each time the robot loses its target, it will induce high reward values to change the synaptic weights. Consequently, the high reward over a longer period of time causes a significant change in the connection values. Shortly before step 28,000, the snake-like robot has learned to follow the target at its first turn, but it still deviates from its optimize course. At this point, we can clearly see the time step in each episode is growing steadily. Afterwards, the controller has successfully learned to follow the target regardless how the target turns. Episodes are only terminated once the robot has reached the maximum steps, therefore, the weights change after step 40,000 are considerably smaller than before and the reward tends to be zero, which is the most ideal situation.

Finally, we load the well trained synaptic weights to the SNN and execute the controller on the training scenario. The deviation of the target in the visual field is shown in Fig. 7. Since the head module wiggles side to side due to its inherent

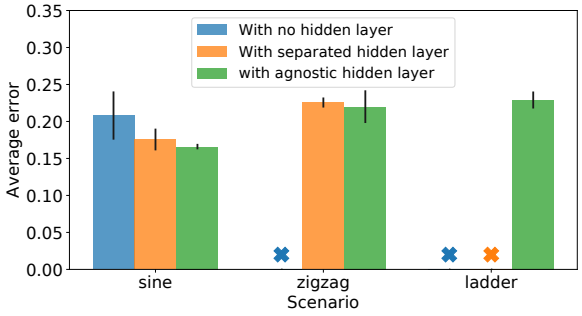


Fig. 11: Tracking accuracy comparison for SNN without hidden layer, SNN with separated hidden layer, and our proposed SNN with agnostic hidden layer.

motion pattern, there is a fluctuation of the distance curve. The tracking accuracy is presented by the distance error in the visual field, which is around 0.124 in this case.

V. Discussion

In this section, the proposed SNN will be performed in three target tracking testing scenarios to inspect the adaptability of the controller. Furthermore, the SNN controller will be compared with two existing SNNs with different topologies in terms of tracking accuracy in testing scenarios.

A. Testing Scenarios for Target Tracking

To examine the adaptability and practicality of the proposed controller, another three tracking trajectories are used to run the SNN controller (See Fig. 2). Meanwhile, these results are shown by plotting the trajectories of the target and robot at the same time. For the first sine-wave trajectory, we have double the frequency as the training scene to enhance the tracking difficulty. From Fig. 8(a) we can see that the robot follows the path of the target steadily, even it doubles its frequency as the training sine track. For the second ladder trajectory in Fig. 8(b), it is even more difficult compared to the first one due to those sharp corners. We can find that the robot follows it closely while sometimes cuts across short path to keep going with the target. For the third triangular trajectory in Fig. 8(c), the robot has difficulty in following the target precisely due to its ordinary maneuverability. However, it keeps cutting short path when the target makes a sharp turn, yielding a sine curve matching the target triangular path.

B. SNN Topology

In order to present the superiority of the proposed algorithm, the performance is further compared to two SNNs with different typologies. As shown in Fig. 9, the first kind SNN architecture has no hidden layers and simply connects the input and output neurons with R-STDP synapses in an all-to-all fashion. The second network has a hidden layer that is separated as such that half of the hidden layer neurons are connected only to the left and half only to the right output neuron, as depicted in Fig. 10.

For the first SNN with no hidden layer, the reward r in (14) is just assigned to all the synapses directly. For the second SNN with separated layer, the reward is not only given to the synapses connecting the output neurons, but also to the synapsed connecting the neurons in the input and hidden layer as well. However, for our proposed SNN, the reward for the synapse between input and hidden layer is assigned according to the rule in (15).

To evaluate the performance of these three SNN controllers, we run them in the testing scenarios to measure their performances in terms of the lateral localization tracking accuracy. All these SNNs have the same training environment and testing parameters to ensure fairness. Specifically, the target position in the visual field of the robot is recorded to depict the tracking accuracy and calculated as (13). If the target is well tracked in the middle, the position value will be zero. On the other hand, if the target shifts to the left or right side, the position value will be recorded as $+/-1$.

Each experiment runs five times and the comparison results are shown in Fig. 11. For performing the controllers in the double-frequency sine trajectory scenario, our proposed SNN with agnostic hidden layer gives not only the lowest error 0.166, but also a small standard deviation 0.0037. The SNN with separated hidden layer also shows better performance than the SNN without hidden layer. The SNN without hidden layer exhibits the largest error 0.208. Unfortunately, the SNN without hidden layer does not manage the zigzag or ladder trajectory scenarios totally. Therefore, it is depicted with the cross marker ✖. In the zigzag scenario, the two SNNs with hidden layer performs similarly error around 0.22, which is higher that the first scenario due to its difficulty. But the one with agnostic hidden layer exhibits higher deviation surprisingly. For the ladder scenario, only the SNN with agnostic hidden layer is capable of finishing it to reach the maximum steps with error 0.229. The separated SNN can manage the task occasionally, therefore, it is also marked as failure.

Generally among all these three SNNs, the one with agnostic hidden layer shows the best performance in the target tracking task with comparatively small deviation and adaptability to unknown scenarios. Explanations for this can be found in the topology of the SNN with agnostic hidden layer that allow for higher robustness due to its all to all synapses.

VI. Conclusion and Outlook

In this paper, we proposed a general approach to back propagate rewards to synapses inside each layer of an SNN on the basis of R-STDP. We took a snake-like robot as an agent to carry out target tracking tasks as case studies. First, our SNN controller is capable of learning the training scenario and working out the unknown scenarios. Second, by comparing to SNNs that simply assign reward to all the synapses, our algorithm shows better performance in terms of accuracy. For future work, the R-STDP learning rule will be investigated by using deep network architectures and implemented in real-life robot applications.

Acknowledgement

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, HBP) and the Chinese Scholarship Council.

REFERENCES

[1] S. J. Thorpe and J. Gautrais, "Rapid visual processing using spike asynchrony," in *Advances in neural information processing systems*, 1997, pp. 901–907.

[2] M. Mattia and P. D. Giudice, "Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses," *Neural Computation*, vol. 12, no. 10, pp. 2305–2329, 2000.

[3] S. Thorpe, A. Delorme, and R. Van Rullen, "Spike-based strategies for rapid processing," *Neural Networks*, vol. 14, no. 6-7, pp. 715–725, 2001.

[4] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, "A survey of robotics control based on learning-inspired spiking neural networks," *Frontiers in Neurorobotics*, vol. 12, p. 35, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2018.00035

[5] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training Deep Spiking Neural Networks using Backpropagation," vol. 10, no. November, pp. 1–10, 2016. [Online]. Available: http://arxiv.org/abs/1608.08782

[6] A. Cyr and M. Boukadoum, "Classical conditioning in different temporal constraints: an stdp learning rule for robots controlled by spiking neural networks," *Adaptive Behavior*, vol. 20, no. 4, pp. 257–272, 2012.

[7] 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.

[8] M. B. Milde, H. Blum, A. Dietmller, D. Sumislawska, J. Conradt, G. Indiveri, and Y. Sandamirskaya, "Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system," *Frontiers in Neurorobotics*, vol. 11, p. 28, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2017.00028

[9] P. Arena, S. D. Fiore, L. Patan, M. Pollino, and C. Ventura, "Insect inspired unsupervised learning for tactic and phobic behavior enhancement in a hybrid robot," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, July 2010, pp. 1–8.

[10] A. Bouganis and M. Shanahan, "Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, July 2010, pp. 1–8.

[11] E. M. Izhikevich, "Solving the distal reward problem through linkage of stdp and dopamine signaling," *Cerebral Cortex*, vol. 17, no. 10, pp. 2443–2452, 2007. [Online]. Available: http://dx.doi.org/10.1093/cercor/bhl152

[12] R. V. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neural Computation*, vol. 19, no. 6, pp. 1468–1502, 2007.

[13] A. Mahadevuni and P. Li, "Navigating mobile robots to target in near shortest time using reinforcement learning with spiking neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 2243–2250.

[14] M. S. Shim and P. Li, "Biologically inspired reinforcement learning for mobile robot collision avoidance," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 3098–3105.

[15] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013*. IEEE, 2013, pp. 1321–1326.

[16] M.-O. Gewaltig and M. Diesmann, "Nest (neural simulation tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[17] S. Ma, "Analysis of snake movement forms for realization of snake-like robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 4, May 1999, pp. 3007–3013 vol.4.

[18] Z. Bing, L. Cheng, G. Chen, F. Röhrbein, K. Huang, and A. Knoll, "Towards autonomous locomotion: Cpg-based control of smooth 3d slithering gait transition of a snake-like robot," *Bioinspiration & biomimetics*, vol. 12, no. 3, p. 035001, 2017.

[19] J. Kaiser, J. C. V. Tieck, C. Hubschneider, P. Wolf, M. Weber, M. Hoff, A. Friedrich, K. Wojtasik, A. Roennau, R. Kohlhaas *et al.*, "Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks," in *IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR),*. IEEE, 2016, pp. 127–134.

[20] Z. Bing, C. Meschede, K. Huang, G. Chen, F. Rohrbein, M. Akl, and A. Knoll, "End to end learning of spiking neural network based on r-stdp for a lane keeping vehicle," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.