2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9341520

# Target Tracking Control of a Wheel-less Snake Robot Based on a Supervised Multi-layered SNN

Zhuangyi Jiang[1] Richard Otto[1] Zhenshan Bing[1] Kai Huang[2,3] and Alois Knoll[1]

***Abstract*— The snake-like robot without wheels is a bio-inspired robot whose high degree of freedom results in a challenge in autonomous locomotion control. The use of a Spiking Neural Network (SNN) which is a biologically plausible artificial neural network can help to achieve the autonomous locomotion behavior of snake robots in an energy-efficient manner. Approaches that use an SNN without hidden layers have been applied in the single-target tracking task. However, due to the complexity of the 3D gaits on a wheel-less snake robot and the imprecision of the pose control while in motion, they have some fluctuation that adversely affects their performances. In this work, we design two multi-layered SNNs with different topology for a wheel-less snake robot to track a certain moving object. The visual signals obtained from a Dynamic Vision Sensor (DVS) are fed into the SNN to drive the locomotion controller. Furthermore, the Reward-modulated Spike-Timing-Dependent Plasticity (R-STDP) learning rule is utilized to train the SNN end-to-end. Compared to the SNN without hidden layers, the proposed multi-layered SNN with a separated hidden layer shows its advantage in terms of robustness.**

## I. INTRODUCTION

With the increase in the demand for special-purpose robots, especially in disaster rescue [1], space exploration [2], and surgery [3], the bio-inspired snake robot has attracted more and more attention. Various snake-like robots have been developed for the purposes as mentioned, such as the snake robot with wheels [4], the wheel-less snake robot [1], and the crawler snake robot [5]. The corresponding locomotion control approaches have been proposed as well, including kinematic modeling [6], CPG controlling [7], etc. However, the autonomous locomotion control of the wheel-less snake robot remains a challenging and poorly solved issue, due to the high degree of freedom and the use of complex 3D gaits.

To perform autonomous locomotion tasks consisting of perception, decision making, and action, Spiking Neural Network (SNN) provides a biologically plausible and energy-efficient model [8]. It has two significant advantages: (1) SNN is good at processing time-dependent patterns because spikes processed and propagated in SNNs carry temporal information. By using SNNs, fewer neurons are needed in the controller than those of the classical neural networks [9]. (2) SNN is able to directly process the visual spikes generated by a DVS. Therefore, it can make full use of the properties of DVS, such as detecting motion with low latency and high dynamic range, to improve the perception ability of a robot in mobile scenarios, especially for target tracking.

There have been numerous literature attempting to achieve autonomous locomotion of a mobile robot by using SNNs. In one of these reports [10], a basic SNN only composed by an input layer and an output layer was applied to autonomously control an arm of a humanoid robot. Cao et al. [11] designed an SNN-based controller to implement the target tracking on an autonomous vehicle robot. Bing et al. [12] proposed a reward assignment algorithm for training an arbitrary multi-layered SNN and achieved autonomous target tracking on a snake robot with passive wheels under a 2D slithering gait. Compared to the 2D slithering gait, the 3D slithering gait of the wheel-less snake robot will reduce the pose stability and enlarge the shift range of the target in the camera view while following a moving target.

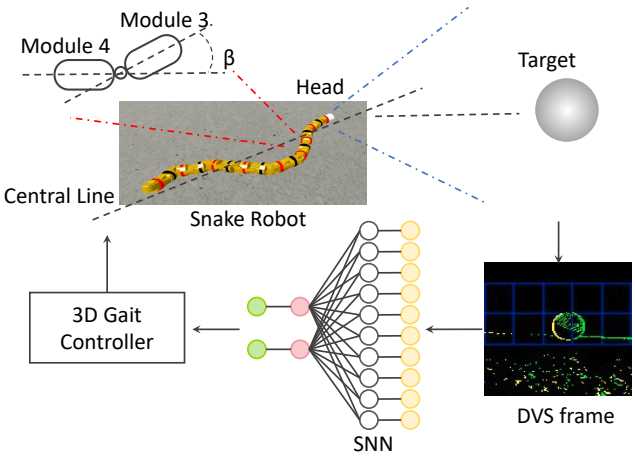



Fig. 1. The framework of the SNN-based target tracking control on a wheel-less snake robot. The accumulated frame of the DVS is encoded into spikes, green points are positive events, yellow points are negative events. The ROI is divided into 10 grids in blue without overlap. After processing by a SNN, output spikes are decoded into control signals for snake robot. The relative angle $\beta$ between two modules is illustrated on the upper-left($\beta = -30°$).

However, researches on the SNN-based autonomous locomotion control of a wheel-less snake robot are still rare. Chen et al. [13] proposed a hybrid neuromorphic computing paradigm and achieved semi-autonomous locomotion of a wheel-less snake robot. In our previous work [14], a static target tracking framework partly based on SNN was proposed for a wheel-less snake robot. The SNN was only used to detect the object so that it is a non-end-to-end model without training.

In this work, we address the issue of SNN-based moving-target tracking on a wheel-less snake robot. Two multi-layered SNNs with different topology are designed to achieve end-to-end target tracking, and further, a Dynamic Vision Sensor (DVS) is utilized to perceive the target and encode it as spikes that are fed into the SNN to drive the locomotion controller of the snake robot. Furthermore, we use the R-STDP learning rule to train a multi-layered SNN with a

[1]Zhuangyi Jiang, Richard Otto, Zhenshan Bing and Alois Knoll are with Department of Informatics, Technical University of Munich, Germany. {jiangz, ottor, bing, knoll}@in.tum.de
[2]Kai Huang is with School of Data and Computer Science, Sun Yat-sen University, China. huangk36@mail.sysu.edu.cn
[3]Kai Huang is also with Peng Cheng Laboratory, China.

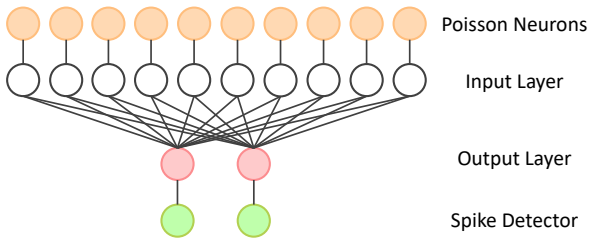



Fig. 2. The topology of the basic SNN.

unified hidden layer and another one with a separated hidden layer, respectively. Finally, a basic SNN without hidden layers is built as a benchmark for evaluating the tracking performance of the proposed multi-layered SNNs.

## II. METHODOLOGY

In the Neurorobotics Platform (NRP) [15][16], we create the framework for target tracking control of a wheel-less snake robot. This section describes the model of the wheel-less snake robot mounted with a DVS, the SNN models, and the R-STDP learning rule.

### A. Simulation Environment

The simulation environment is built in NRP where a target sphere with a radius of 30cm is spawned 1.6m in front of a wheel-less snake robot, as shown in Fig. 1. The sphere moves along a predefined trajectory, while the target moves a little bit faster than the snake robot, but not further away from the snake robot than the initial distance.

*1) Wheel-less Snake Robot:* The wheel-less snake robot used in this work has 17 modules that are connected alternatively in dorsal and in lateral directions by using 1-DoF revolute joints. The first module, namely the head module, is a hollow cylinder where a DVS sensor is mounted. Unlike a biological snake that can continuously curve its body, the snake robot creates a 3D shape by rotating its joints along a fixed axis in the range of $[-90°, 90°]$. This allows the snake robot to perform 3D gaits inspired by motions of the biological snake, such as slithering, rolling, and sidewinding.

In this work, the wheel-less snake robot moves under the slithering gait [17] which is described in the equation:

$$\alpha_i = \begin{cases} P_i A_o \sin(i\Omega_o + t\omega_o + \delta_o) + C_o, & i\ is\ odd \\ P_i A_e \sin(i\Omega_e + t\omega_e) + C_e, & i\ is\ even \end{cases} \quad (1)$$

where $\alpha_i$ is the joint position of the i-th joint ($i \in [1, 16]$), $P$ is the linear dependency, $A$ is the amplitude, $\Omega$ is the spatial frequency, $\omega$ is the temporal frequency, $\delta$ is the phase difference, and $C$ is the body offset, the subscript $o$ indicates the horizontal joint, $e$ indicates the vertical joint. The robot is easily controlled in which direction it should go by just adjusting $C_o$. In order to keep the orientation of the head module parallel to the movement direction, a compensation $C_{compensation}$ is applied to the first joint so that the DVS can always sense the environment in front of the snake robot while moving. As a result, objects in front of the robot will periodically change their position in the horizontal direction throughout the movement.

*2) Neuromorphic Vision Sensor:* The Dynamic Vision Sensor (DVS) is a neuromorphic vision sensor, also known as an event-based camera [18]. In contrast to frame-based sensors, DVS outputs asynchronously events that respond to the change in brightness in the order of microseconds. For each pixel, only if the difference of brightness exceeds a certain threshold, a spike is emitted. The spikes are encoded as Address Event Representations (AERs), that is, each spike can be represented as a tuple $(x, y, t, p)$. $(x, y)$ is the position of the triggered pixel, t is the timestamp, and p is the polarity of event, +1 indicates the positive spike while -1 indicates the negative one. A simulated DVS with $128 \times 128$ pixels was applied in this work.

### B. Basic SNN Architecture

The purpose of building a basic SNN is to set up a benchmark for evaluating the performance of multi-layered SNNs. Getting inspired by [10][19], we design an SNN without hidden layers for our wheel-less snake robot as shown in Fig. 2. This basic SNN consists of ten Poisson neurons as spike encoders, ten LIF input neurons and two LIF output neurons, and two more spike detectors. Input neurons and output neurons are fully connected by R-STDP synapses, while Poisson neurons and spike detectors are connected to the main body by one-to-one connections with static synapses. A Poisson neuron is regarded as a spike generator that transforms numerical values into spike trains with a proportional firing rate to drive an input neuron. In order to parrot these spikes to pass through the input layer, we set the refractory period of input LIF neurons to zero and set their threshold voltage close to the resting voltage. Finally, spike detectors collect the spikes generated by output neurons to spike decoders directly connecting with the robot controller.

### C. Multi-layered SNN Architecture

Different from SNNs without hidden layers, a multi-layered SNN is composed of an input layer, an output layer, and at least one hidden layer. We designed two multi-layered SNNs as shown in Fig. 3 and Fig. 4, respectively. Both of them have a separated input layer and a separated output layer, aiming to control the head module and other modules in two phases. Ten input neurons responsible for processing DVS spikes and obtaining the position of the target in the image plane are grouped into the first layer. Six input neurons used for gathering the joint positions and two output neurons make up the second layer. The output neurons in the second layer are responsible for controlling the head of the snake robot to face the target, while the output neurons in the last layer are responsible for controlling the movement direction of the snake robot body to follow the target. The idea behind this is the processing of the data from different sensors in a hierarchical order before fusing in the hidden layer.

The difference between two multi-layered SNNs is whether it has a unified hidden layer. As shown in Fig. 3, its hidden layer is unified and consists of eight neurons. Neurons in the hidden layer are fully connected to output neurons.

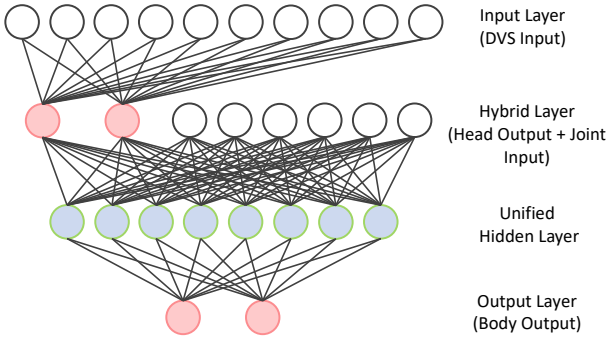



Fig. 3. The topology of the multi-layered SNN with a unified hidden layer.

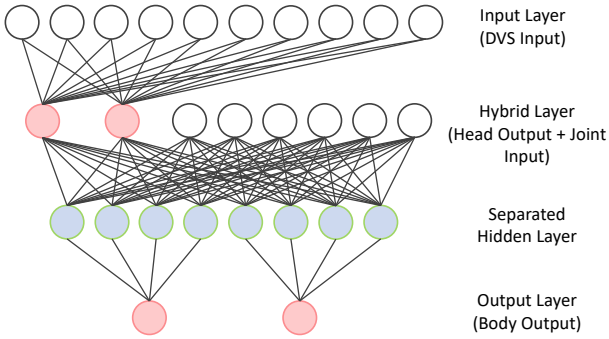



Fig. 4. The topology of the multi-layered SNN with a separated hidden layer.

In contrast, the SNN as shown in Fig. 4 has a separated hidden layer. Neurons in the left part of the hidden layer are connected to the left output neuron while neurons in the right side are connected to the right output neuron. The separated hidden layer means less interaction between two output neurons.

### *D. Data Encoding and Decoding*

In order to drive an SNN, data from sensors need to be encoded as spikes which can be understood by spiking neurons, and control signals need to be decoded as values to control motors. All data is updated at a fixed simulating timestep that is 20ms in our case.

Different from the real DVS, the DVS in NRP outputs the significant difference of intention between two consecutive image frames so that all events occur at the same time. We first select a ROI and then divided it into 2×5 grids, as shown in Fig. 1. We use the grid with five columns so that there is a column in the middle of the view. The major benefit is reducing the fluctuation of the control signal when the target is right in front of the snake robot. Each grid is corresponding to a Poisson neuron. We set the firing rate of Poisson neurons to a certain frequency that is the product of the number of spikes $N_{event}$ occurred in each grid and the frequency of data updating. Hence, the firing rate f is calculated by

$$f = N_{event}/t_{step} \tag{2}$$

Moreover, three pairs of input neurons in the second layer of the multi-layer SNN are used to encode the joint position of the first, third, and fifth joint, respectively. According to Eq. 1, these horizontal joints can provide information about the pose of the snake robot in the horizontal plane under the slithering gait. Each joint position is encoded by a pair of Poisson neurons, one for the positive angle and the other for the negative angle. It means that only one Poisson neuron is active at a time so that the firing rate of a pair of Poisson neurons is given as follows:

$$f_P = \max(0, k\alpha), \;\; f_N = \max(0, -k\alpha) \tag{3}$$

where $f_P$ represents the firing rate of the Poisson neuron for the positive angle, while $f_N$ is that for the negative angle. $\alpha$ is the joint position and $k$ is a positive coefficient. This can be interpreted as the neuron activity, releasing the antagonist and the agonist simultaneously. Taking Fig. 1 as an example, the joint position $\beta$, the rotating angle of module 3 to module 4, is negative so that $f_P$ is 0 and $f_N$ is positive.

For decoding, the output of the network is represented by a pair of neurons as well. The difference of the spike number between two output neurons in each simulation time period is decoded as the turn direction, then normalized by using the total number of spikes occurred on both neurons, which is shown as follows:

$$D_{norm} = (N_r - N_l)/(N_r + N_l), D_{norm} \in [-1, 1] \tag{4}$$

where $N_r$ is the spike number of the right neuron and $N_l$ is the spike number of the left one, $D_{norm}$ represents the intention of changing the current direction.

In order to avoid bursts of the turn direction, we apply a filter to smooth it. $D_t$, the final turn direction, is given by

$$D_t = \min(1, \max(-1, (D_{t-1} + \eta D_{norm}))) \tag{5}$$

Where $\eta$ is a constant coefficient that defines how aggressively the direction can be changed. $D_t$ is always in the range of -1 to 1, which can be projected to a turning angle as the robot has a finite steering range.

### *E. Reward function for R-STDP learning*

*1) Reward-modulated Spike-Timing-Dependent-Plasticity:* Reward-modulated spike-timing-dependent-plasticity is derived from neuroscience studies. In this model, the weight change of a synapse is calculated by the eligibility trace of STDP and a supervised signal $a_{reward}$ from a dopamine neuron. The mechanism of R-STDP can be described by the following equations [20][21][22].

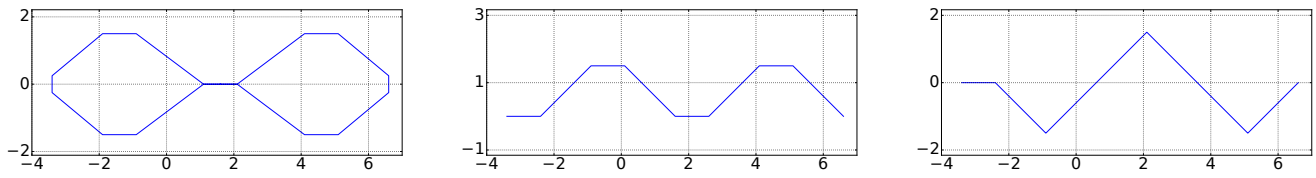
The eligibility trace of STDP is defined by

$$\dot{c} = -\frac{c}{\tau_c} + STDP(\Delta t)\delta(t - s_{pre/post}) \cdot C_1 \tag{6}$$

where $\tau_c$ is the time constant, $\delta(t)$ is the Dirac delta function, $s_{pre/post}$ is the time when a pair of pre- and post-synaptic spikes occurs, and $C_1$ is a constant coefficient. Only when the pre-synaptic spike occurs before the post-synaptic one within a time window, will the eligibility of the STDP increase. Then, the dynamics of the reward are described by

$$\dot{n} = -\frac{n}{\tau_n} + \frac{\delta(t - s_n)}{\tau_n} \cdot C_2 \tag{7}$$

where $n$ is the neuromodulator concentration, $\tau_n$ is the time constant, $s_n$ is the release time of the neuromodulator, and



(a) 8-like track. (b) Ladder shape track. (c) Zigzag track.

Fig. 5. Tracks of the target. (a) is for training, (b) and (c) are for testing. The wheel-less snake robot starts moving at point (-5.0, 0.0), and when each training episode finished, the track will be axisymmetric about the x-axis.

$C_2$ is a constant coefficient. If $b$ is defined as the baseline concentration of the neuromodulator, the weight change is then given by

$$\dot{w} = c \cdot (n - b) \tag{8}$$

*2) Reward Assignment:* Finding a suitable reward function is crucial for the successful training of SNNs. In this work, rewards for output neurons are defined by the angle between the moving direction of the robot and the target:

$$r_{right} = -r_{left} = a_{target} \cdot C_{reward} \tag{9}$$

where $r_{left}$ and $r_{right}$ are the rewards for the left and right output neuron, respectively, $a_{target}$ is the angle to the target, and $C_{reward}$ is a positive coefficient. In other words, synapses connected to the left neuron get the same positive reward when $a_{target}$ is negative, and vice versa. It means the neuron on the same side as the target will get rewarded while the other side will get punished.

Moreover, to propagate rewards to the hidden layers of SNNs, a method similar to the back-propagation algorithm used in the previous study [12] is also employed. The reward for the $j$-th neuron in the $i$-th layer counted from the output layer is given by

$$r_{i,j} = \frac{\sum_{k=1}^{Y_i \ \ 1} (r_{i-1,k} \cdot w_{i,j,k})}{\max\left(|w_{i,j,1}|, |w_{i,j,2}|, \ldots, |w_{i,j,Y_i \ \ 1}|\right) \cdot Y_{i-1}} \tag{10}$$

where $Y_{i-1}$ is the number of neurons in the $(i-1)$-th layer, $r_{i-1,k}$ is the reward of the $k$-th neuron in the $(i-1)$-th layer, $w_{i,j,k}$ is the weight of the synapse that connects the $k$-th neuron of the $(i-1)$-th layer and the $j$-th neuron of the $i$-th layer. We take the SNN shown in Fig. 3 as an example, each neuron in the hidden layer has two outgoing synapses. Thus, the reward for the hidden layer is calculated by

$$r_h = \frac{r_l \cdot w_l + r_r \cdot w_r}{\max(|w_l|, |w_r|) \cdot 2} \tag{11}$$

where $r_l$ and $r_r$ are the rewards for the left and right neuron, respectively, $w_l$ and $w_r$ are the weights of the synapse connected to the left and right output neuron, respectively.

## III. EXPERIMENTS

Experiments for moving-target tracking were conducted in the Neurorobotics Platform (NRP). This section describes differences between experimental setups and shows the results of the training phase. To train the networks, an 8-like path of the target is used as shown in Fig. 5(a). The wheel-less snake robot would be trained by several episodes in which the robot would follow the target moving alternatively in clockwise and counterclockwise.

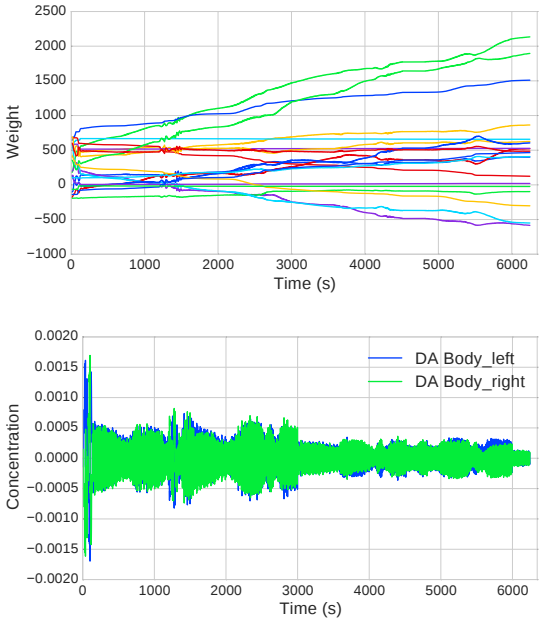



Fig. 6. The changes in weights of the basic SNN as the dopamine concentration (reward) decreases during training.

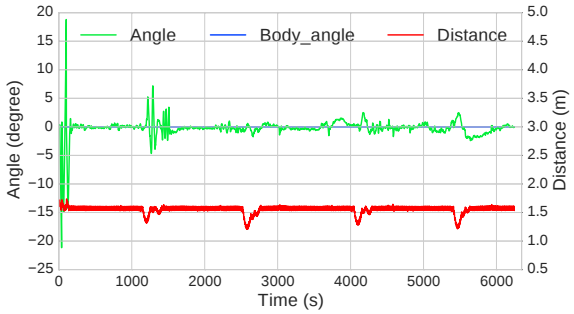
### A. Benchmark Experiment

The basic SNN described in Sec. II-B was trained first as a benchmark for testing performances of the proposed multi-layered SNNs. The training for the first episode finished at the 151093 simulation steps, for a period of 3021.86 seconds, which shows the ability of the basic network in completing the tracking task. Althouth the reward signals are decreasing, the weights were still changing strongly. After completing the second episode in the symmetric track, the training process stopped after the weights gradually getting stabilized. Fig. 7(a) shows the weight changes over the whole training period, the rewards given to the output of the network, and the mean angle and distance to the target over time.

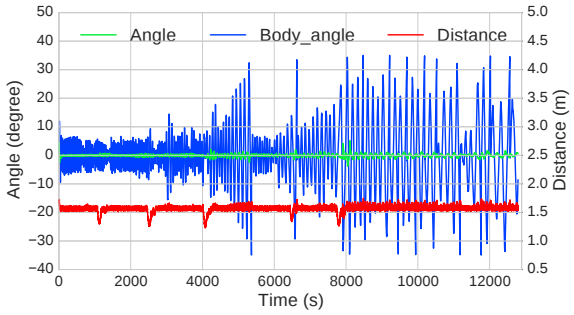
### B. Basic SNN with Head Controller

To avoid failures that could occur when the target moves out of the field of view, we added a head facing component into the basic SNN. The new network would control the moving direction and also the facing direction of the head module of the snake robot. Thus, an additional pair of output neurons is added to steer the head in a similar architecture to the other two output neurons in Fig. 2.

For this setup, the factor $\eta$ described in the Sec. II-D is 0.02 for the head and 0.07 for the body, while the factor




(a) The basic SNN




(b) The basic SNN with Head Controller

Fig. 7. The performances of the basic networks in training. The green curve is the mean angle of the head, the blue one is the mean angle of the body, and the red one is the distance to the ball. The angle is the same as the Body_angle in case (a).

$C_{reward}$ is 0.0004 for the head and 0.0001 for the body. The reward assigned to the head controller corresponds to the angle of the head to the target. For the body controller, the reward is proportional to the angle between the body and the target which is calculated by

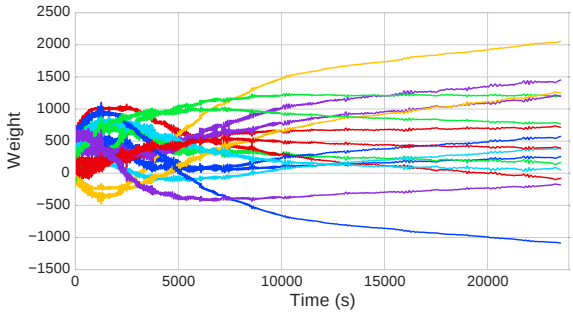
$$a_{body} = a_{head} - d_{head} \tag{12}$$

where $a_{head}$ and $a_{body}$ are the angles the head and the body make to the target, respectively, $d_{head}$ is the compensation related to the output of head controller to make the head of the snake robot face the target. In this case, the robot gets reset if the absolute value of either $a_{head}$ is bigger than 40° or the average of $a_{body}$ is greater than 35° within 625 simulation steps, which is a movement period of the snake robot to move its head from one end to the other and then move back.

As shown in Fig. 7(b), this setup performs quite well at the beginning because the rewards change weights strongly enough and the target moves along a straight line away from the snake robot. However, when the target moves in the ladder path, the network gets more and more unstable. The training stops several times resulting from the failure occurs shortly after the start of the episode. It suggests that either the network is not complex enough or the information for the input of the SNN is not sufficient.

### C. Multi-layered SNNs

The above experiments suggest that a basic network lacks the complexity to solve this task, therefore two kinds of multi-layered SNN were built. Since the head controller



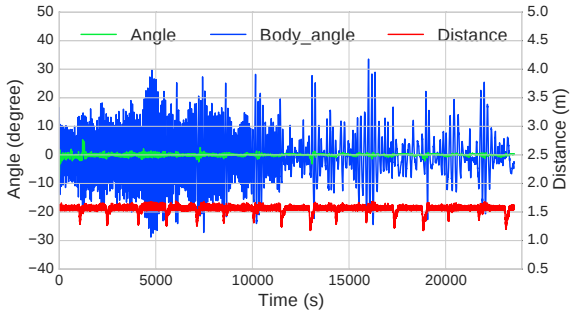




Fig. 8. The weight changes of the synapse connected to the body output of SNN with a unified hidden layer and the performance of the SNN with a unified hidden layer in training.

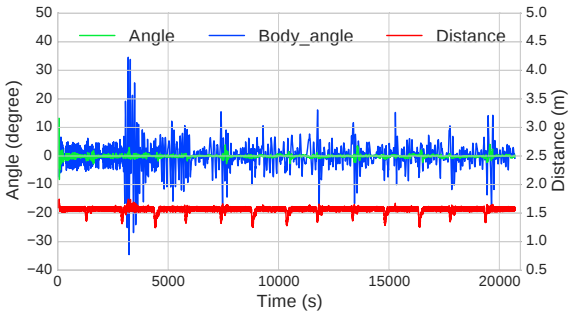



Fig. 9. Performance of the SNN with a separated hidden layer in training.

worked well, the head output neurons would still be utilized. These two output neurons and six neurons in charge of encoding the joint angles make up the hidden layer. Head output neurons only use the visual signals to estimate the target direction in the image plane, then the body output neurons utilize both the target direction and the angles of the partial joints to adjust the pose of the robot body.

*1) Unified Hidden Layer:* For synapses connected to the hidden layer, we used a larger $\tau_c$ to accumulate the STDP eligibility over a larger time span. Consequently, the synapse remembers information for a long time, which can trigger the post-synapse neuron more frequently. On the other hand, both the LTP and LTD window of STDP were scaled down by the same factor $\tau_c$ to avoid drastic changes in the weights.

For this setup, the training phase was much longer than the basic SNN. Fig. 8(a) shows the weight changes lessen over time in this setup. As shown in Fig. 8(b), both the head angle and the mean body angle with peaks of around 25° get stable at the end of the training.

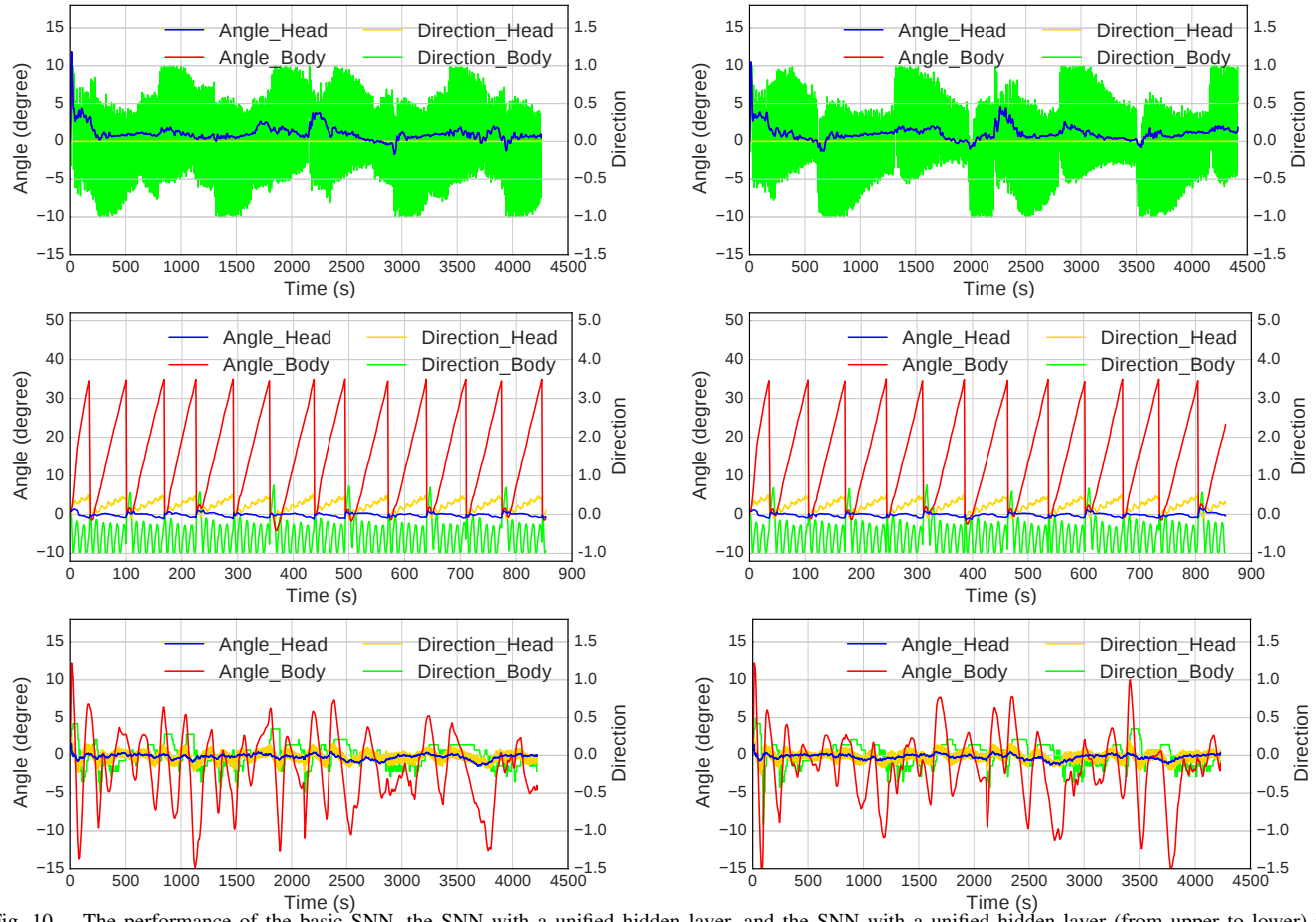



Fig. 10. The performance of the basic SNN, the SNN with a unified hidden layer, and the SNN with a unified hidden layer (from upper to lower) in testing. The left column is the results testing on the ladder track, and the right colums are the results testing on the zigzag track. Angle_Head is the real angle between the head of the snake robot and the target, Direction_Head and Direction_Body are the values decoded by the head output neuron and the body output neuron, respectively.

*2) Separated Hidden Layer:* We trained the SNN with a separated hidden layer by using almost the same parameters as that one with a unified hidden layer. As shown in Fig. 4, the hidden layer was split into two groups with four neurons each and all outgoing connections from a group connect to only one output neuron. Hence, Eq. 11 can be simplified to

$$r_{h_r} = r_{o_r} \cdot w_r/|w_r|,\ r_{h_l} = -r_{o_l} \cdot w_l/|w_l| \qquad (13)$$

where $r_{h_r}$ and $r_{h_l}$ are the rewards for the hidden layer connected to the right output and the left output, respectively. $r_{o_r}$ and $r_{o_l}$ are the rewards for the right output neuron and the left output neuron, respectively. $w_l$ is the weight of the connection from a neuron in the left hidden layer to its output, and $w_r$ is the weight of the connection from a neuron in the right hidden layer to its output. This simplification is the result of hidden neurons influencing only one body output neuron. Training this SNN is also time-consuming and the overall performance is comparable as shown in Fig. 9.

## IV. TESTING RESULT

To test the performance of the trained networks two different paths for the target are designed as shown in Fig. 5(b), they have a similar curvature as the training path. Both the original paths and the symmetric paths are tested.

### A. Performance

Fig. 10 illustrates the performances of the basic SNN, the SNN with a unified hidden layer, and the SNN with a separated hidden layer, respectively. Charts in the left column were tested on the ladder track while those on the right side were tested on the zigzag track. First, the basic SNN finished the test without any failure, even when the path was mirrored on the horizontal axis. The mean angle between the head of the snake robot and the target ball rarely surpasses an absolute value of 4°. In this case, the head angle and the body angle are almost the same. The direction of the head which is the value decoded by the head output neurons is zero, but the direction of the body reaches the range limitation several times. It means that the parameter for the gait controller should be adjusted frequently to move towards the target. Then, the network with a unified hidden layer was not able to perform at all and never made it to the first episode. The simulation was reset in every 50s when the SNN failed due to the very high trend to go right of the snake robot. Finally, the SNN with a separated hidden layer performed well and kept the absolute mean angle of the head below 2°. Although the body angle was changing in a larger range than the basic SNN, the directions responsible for adjusting the parameters for the gaits of the snake robot dropped dramatically. It

means that we can use fewer control signals with smaller parameter change to achieve autonomous target following on the wheel-less snake robot by using the proposed SNN with a separated hidden layer.

### B. Discussion

The results show that the basic network without hidden layers is capable of navigating the snake robot to follow the target, but it would fail when the target moves out of the field of view. It cannot perform well when adding the head controller to solve the target missing problem. After adding the head controller to keep the head always facing the target, the DVS input was found to be insufficient to make output neurons excited. One reason might be the reduction of generated events on account of less relative motion between the head module and the target. Another reason is, the target may not be in front of the robot when the target appears in the central area of the DVS view. Therefore, additional information, like the head joint position, has to be fused in to distinguish the relative position of the target.

Nevertheless, the basic SNN with two kinds of output neuron still performed badly when adding additional inputs from joint encoders. In this more complex setting, neither the DVS input nor the head joint positions can directly derive an action on their own. Only with more information, like the angle of other joints, and more complex network, a valid decision can be made. This implicit the relationship between so many inputs should be described by a more complex network. This would also explain why none of the basic networks with a head controller were stable.

More surprising was the bad performance of the SNN with a unified hidden layer. Even though the weights have been stable, it cannot control the robot. A possible reason is that all synapses connecting the hidden layer to one output neuron shared the reward and the synapses for left and right output neurons had opposite polarity. Therefore, the reward signals might weaken each other. The SNN with a separated hidden layer could learn faster, as the weights of the hidden layer were only adjusted for one neuron. It also performed better with less direction fluctuations than the basic SNN.

## V. CONCLUSIONS

In this paper, we design two multi-layered SNNs to achieve the moving-target tracking control on a wheel-less snake robot. The simulation experiments conducted in the NRP demonstrated that the multi-layered SNN with a separated hidden layer can achieve better tracking performance than a basic SNN without hidden layers. Compared to the basic SNN, the relative direction of the target to the robot is with less fluctuation when using the multi-layered SNN with a separated hidden layer, which means higher robustness and less risk in missing target while the robot following.

## ACKNOWLEDGMENT

This research has received funding from the European Unions Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2). This work is supported in part by the scholarship from China Scholarship Council (CSC) under the Grant No. 201606270201 and the funding from Science and Technology Planning Project of Guangzhou city of China under Grant 202007050004.

## REFERENCES

[1] J. Whitman, N. Zevallos, M. Travers, and H. Choset, "Snake robot urban search after the 2017 mexico city earthquake," in *2018 IEEE international symposium on safety, security, and rescue robotics (SSRR)*. IEEE, 2018, pp. 1–6.

[2] M. D. Hancher and G. S. Hornby, "A modular robotic system with applications to space exploration," in *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*. IEEE, 2006, pp. 8–pp.

[3] P. Gomes, "Surgical robotics: Reviewing the past, analysing the present, imagining the future," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 2, pp. 261–266, 2011.

[4] L. Pfotzer, S. Klemm, A. Rönnau, J. M. Zöllner, and R. Dillmann, "Autonomous navigation for reconfigurable snake-like robots in challenging, unknown environments," *Robotics and Autonomous Systems*, vol. 89, pp. 123–135, 2017.

[5] T. Kamegawa, T. Yarnasaki, H. Igarashi, and F. Matsuno, "Development of the snake-like rescue robot" kohga"," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 5081–5086.

[6] F. Matsuno and K. Suenaga, "Control of redundant 3d snake robot based on kinematic model," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 2061–2066.

[7] X. Wu and S. Ma, "Cpg-based control of serpentine locomotion of a snake-like robot," *Mechatronics*, vol. 20, no. 2, pp. 326–334, 2010.

[8] 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.

[9] X. Wang, Z.-G. Hou, A. Zou, M. Tan, and L. Cheng, "A behavior controller based on spiking neural networks for mobile robots," *Neurocomputing*, vol. 71, no. 4-6, pp. 655–666, 2008.

[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)*. IEEE, 2010, pp. 1–8.

[11] Z. Cao, L. Cheng, C. Zhou, N. Gu, X. Wang, and M. Tan, "Spiking neural network-based target tracking control for autonomous mobile robots," *Neural Computing and Applications*, vol. 26, no. 8, pp. 1839–1847, 2015.

[12] Z. Bing, Z. Jiang, L. Cheng, C. Cai, K. Huang, and A. Knoll, "End to end learning of a multi-layered snn based on r-stdp for a target tracking snake-like robot," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9645–9651.

[13] G. Chen, Z. Bing, F. Röhrbein, J. Conradt, K. Huang, L. Cheng, Z. Jiang, and A. Knoll, "Toward brain-inspired learning with the neuromorphic snake-like robot and the neurorobotic platform," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 1, pp. 1–12, 2017.

[14] Z. Jiang, Z. Bing, K. Huang, and A. C. Knoll, "Retina-based pipe-like object tracking implemented through spiking neural network on a snake robot," *Frontiers in neurorobotics*, vol. 13, p. 29, 2019.

[15] F. Roehrbein, M.-O. Gewaltig, C. Laschi, G. Klinker, P. Levi, and A. Knoll, "The neurorobotic platform: a simulation environment for brain-inspired robotics," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*. VDE, 2016, pp. 1–6.

[16] E. Falotico, L. Vannucci, A. Ambrosano, U. Albanese, S. Ulbrich, J. C. Vasquez 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, p. 2, 2017.

[17] Z. Bing, L. Cheng, K. Huang, Z. Jiang, G. Chen, F. Röhrbein, and A. Knoll, "Towards autonomous locomotion: Slithering gait design of a snake-like robot for target observation and tracking," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2698–2703.

[18] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 $\times$ 128 120 db 15$\mu$s latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[19] Z. Bing, C. Meschede, G. Chen, A. Knoll, and K. Huang, "Indirect and direct training of spiking neural networks for end-to-end control of a lane-keeping vehicle," *Neural Networks*, vol. 121, pp. 21–36, 2020.

[20] 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.

[21] W. Potjans, A. Morrison, and M. Diesmann, "Enabling functional neural circuit simulations with distributed computing of neuromodulated plasticity," *Frontiers in computational neuroscience*, vol. 4, p. 141, 2010.

[22] R. Legenstein, D. Pecevski, and W. Maass, "A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback," *PLoS computational biology*, vol. 4, no. 10, 2008.