2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

End to End Learning of Spiking Neural Network based on R-STDP for a Lane Keeping Vehicle

Zhenshan Bing[1], Claus Meschede[1], Kai Huang[2], Guang Chen[3],
Florian Röhrbein[1], Mahmoud Akl[1], and Alois Knoll[1]

Abstract— **Learning-based methods have demonstrated clear advantages in controlling robot tasks, such as the information fusion abilities, strong robustness, and high accuracy. Meanwhile, the on-board systems of robots have limited computation and energy resources, which are contradictory with state-of-the-art learning approaches. They are either too lightweight to solve complex problems or too heavyweight to be used for mobile applications. On the other hand, training spiking neural networks (SNNs) with biological plausibility has great potentials of performing fast computation and energy efficiency. However, the lack of effective learning rules for SNNs impedes their wide usage in mobile robot applications. This paper addresses the problem by introducing an end to end learning approach of spiking neural networks for a lane keeping vehicle. We consider the reward-modulated spike-timing-dependent-plasticity (R-STDP) as a promising solution in training SNNs, since it combines the advantages of both reinforcement learning and the well-known STDP. We test our approach in three scenarios that a Pioneer robot is controlled to keep lanes based on an SNN. Specifically, the lane information is encoded by the event data from a neuromorphic vision sensor. The SNN is constructed using R-STDP synapses in an all-to-all fashion. We demonstrate the advantages of our approach in terms of the lateral localization accuracy by comparing with other state-of-the-art learning algorithms based on SNNs.**

I. Introduction

Pursuing robots to perform complex tasks autonomously has become a realistic prospect, e.g. in the form of self-driving vehicles, space exploration, and collaborative industrial robots. To acquire this high-level intelligence and operate within the real world, robots need to perceive their environment via sensors, which typically deliver high-dimensional data. Today deep network architecture has became a possible solution, since its superiority for extracting highly non-linear functions from training data. However, the high computational demands of deep networks still take a toll, since training them is time consuming, energy-intensive, and typically produces high response delay. In self-driving cars for example, the overall computation consumes a few thousand Watts compared to the human brain, which only needs around 20 Watts of Power [1]. These are considerable disadvantages, especially in mobile applications where real-time responses are important and energy supply is limited.

Authors' Affiliation: [1] Technical University of Munich, Department of Informatics, [2] Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, School of Data and Computer Science, Sun Yat-Sen University. [3] School of Automotive Engineering, Tongji University
Email: bing@in.tum.de, claus.meschede@tum.de, huangk36@mail.sysu.edu.cn, 18503@tongji.edu.cn, florian.roehrbein@in.tum.de, mahmoud.akl@tum.de, knoll@in.tum.de

Fig. 1: Robot task: Lane keeping.

A promising solution to these drawbacks could be given by event-based spiking neural networks that mimic the underlying mechanisms of the brain much more realistically. In nature, information is usually processed using impulses or spikes, making seemingly simple organisms able to perceive and act in the real world exceptionally well and outperform state-of-the-art robots in almost every aspect of life [2]. For example, 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 10 synaptic stages from the retina to the temporal lobe [3]. Therefore, SNNs have tremendous potential to process information more efficiently both in terms of accuracy and speed.

On the other hand, training these kinds of networks is notoriously difficult. The error back-propagation mechanisms commonly used in conventional neural networks can not be directly transferred to SNNs due to the non-differentiabilities at spike times. Therefore, there has been a void of practical learning rules to train SNNs [4]. Initially, SNN-based control tasks were done by manually setting network weights, e.g. in [5], [6], and [7]. Although this approach is able to solve simple behavioral tasks, such as wall following [8] or lane keeping [9], it is only feasible for lightweight networks with few connections. On the level of single synapses, experiments have shown that the precise timing of pre and post-synaptic spikes seems to play a crucial part in the change of synaptic efficacy [10]. With this Spike-Timing-Dependent-Plasticity (STDP) learning rule, networks have been trained in various tasks. For example, Wang constructed a single-layer SNN using proximity sensor data as conditioned stimulus input was then trained in tasks such as obstacle avoidance and target reaching [11, 12]. However, it is still not clear how the brain assigns credit as efficiently as back-propagation does, even some preliminary research tries to bridge the gap by combining back-propagation with SNNs [13, 4].

After that, some research has been done trying to

978-1-5386-3081-5/18/$31.00 ©2018 IEEE

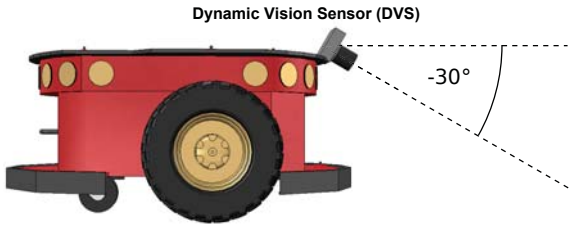
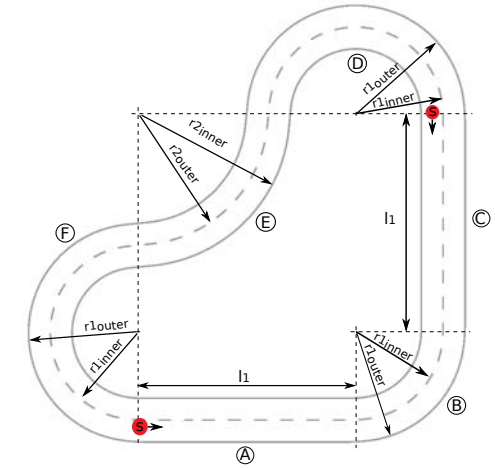


Fig. 2: Pioneer P3-DX robot with dynamic vision sensor (DVS).

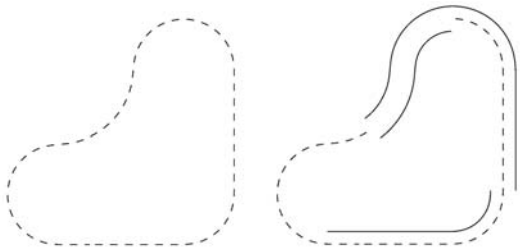
implement biologically plausible reinforcement learning algorithms based on experimental findings in SNNs. Reward-modulated spike-timing-dependent-plasticity (R-STDP) [14][15], which is a learning rule that incorporates a global reward signal in combination with STDP, is recently raised. This approach intends to mimic the functionalities of those neuromodulators, which are chemicals emitted in human brain, e.g. dopamine. Therefore, R-STDP can be very useful for robot control, because it might simplify the requirements of an external training signal and leads to more complex tasks.

However, practical robotic implementations based on R-STDP are rarely found due to its complexity in feeding sensor data into SNNs, constructing and assigning the reward to neurons, and training the SNNs. Specifically, typical sensor data is time-based, such as proximity sensor and conventional vision sensor, rather than event or spike-based. In order to feed the data into a SNN, it has to be converted into spikes somehow. In addition, the reward should be carefully assigned to the SNN, either too high or too low value will both get the learning instable. The network weights are critical for learning as well, otherwise the learning process will consume more time or even cause failures.

To this end, our paper looks to explore the SNNs training algorithms based on R-STDP learning rule and implement them for end to end control in robotics domain. Our main contributions are summarized as follows. First, a simulated lane environment is constructed and adapted with different lane patterns for evaluating algorithms, in which a Pioneer robot mounted with a dynamic vision sensor (DVS) is deployed to generate visual spikes directly. Second, we propose an event-based neural network which use the DVS [16] as the input and calculate the motor commands as the output for a lane keeping task. Instead of manually setting weights or connections like [9, 17], all the neurons in our network are fully connected with R-STDP synapses and the network is trained directly to learn the synaptic weights all by itself. The reward given to the SNN is defined for each motor individually as a linear function of the lane center distance. The network is implemented in NEST using the STDP dopamine synapse model and trained with the rewards calculated from the distance between the robot and lanes. Finally, simulation results of our event-based neural network are analyzed to demonstrate the feasibility to different lanes, and compared with a Braitenberg vehicle controller [9] to



(a) Scenario 1



(b) Scenario 2 (c) Scenario 3

Fig. 3: Different lane keeping scenarios. (a) Scenario 1: The simple lane-keeping scenario consists of a road with 2 lanes and 6 different sections A, B, C, D, E and F. Starting positions are marked with s. Dimensions: $r1_{inner} = 1.75\ m$, $r2_{inner} = 3.25\ m$, $r1_{outer} = 2.25\ m$, $r2_{outer} = 2.75\ m$, $l_1 = 5.0\ m$. (b) Scenario 2: Single lane pattern without boundaries. (c) Scenario 3: Lanes with two different patterns.

embody the accuracy superiority in terms of the deviation of the robot from the center line.

The rest of this paper is structured as follows: Section II describes the simulation environment for the lane keeping tasks. Section III presents the architecture of the SNN and the training results. In Section IV, the simulation results are analyzed and compared to other algorithms. Section V concludes this paper.

II. Right Lane Keeping Tasks

In order to provide a simple and flexible environment to test and compare different algorithms, simulated right lane keeping tasks with different lane patterns for a pioneer robot [18] are set up as case studies (See Fig. 1)

Instead of using the on-board ultrasonic sensors, a DVS camera is attached to the front of the robot with a 30° depression angle as shown in Fig. 2. For further validating the effectiveness and adaptability of the proposed approach, three scenarios with different lane patterns (See Fig. 3). The first scenario in Fig. 3a consists of a circular course with a two-lane road. The road is comprised of two solid lines and a uniformly dashed line in the middle. From the starting position onwards, the outer lane can be divided into

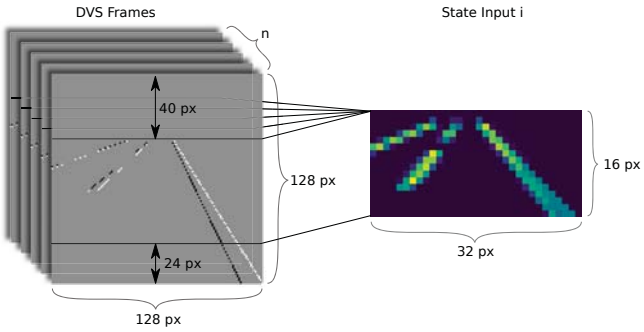


Fig. 4: Conversion of consecutive DVS frames into state input for reinforcement learning. This is done by dividing the original 128×128 DVS frames into small 4×4 regions and counting every event over consecutive frames regardless of increase or decrease illumination. Furthermore, the image is cropped at the top and bottom resulting in a 32×16 image.

six sections, namely, (A) *straight*, (B) *left*, (C) *straight*, (D) *left*, (E) *right*, (F) *left*. During each episode in the training, the robot will switch the start position and moving direction between inner and outer lane at each reset. Therefore, it will experience both left and right turns equally and with different radii as well.

Based on the same layout and dimensions, a second scenario has been implemented testing the algorithms on a different road pattern where the left and right solid lines are missing (See Fig. 3b). In a third scenario, two different road patterns have to be learned in parallel (See Fig. 3c).

III. Event-based SNN Controller

In this section, the SNN controller is construed and trained for steering the robot in aforementioned lane keeping tasks.

A. R-STDP Learning Rule

As the most important theory in neuroscience explaining the adaption of synaptic efficacies in the brain during the learning process, the Spike-Timing-Dependent-Plasticity (STDP) learning rule [19] has been successfully proven by neuroscience experiments [20, 21].

For this work, the weight update rule under STDP as a function of the time difference between pre- and postsynaptic spikes is defined as

$$\Delta t = t_{post} - t_{pre} \tag{1}$$

$$W(\Delta t) = \begin{cases} A_+ e^{-\Delta t/\tau_+}, & \text{if } \Delta t \geq 0 \\ -A_- e^{\Delta t/\tau_-}, & \text{if } \Delta t < 0 \end{cases} \tag{2}$$

$$\Delta w = \sum_{t_{pre}} \sum_{t_{post}} W(\Delta t) \tag{3}$$

, 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_- representing 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.

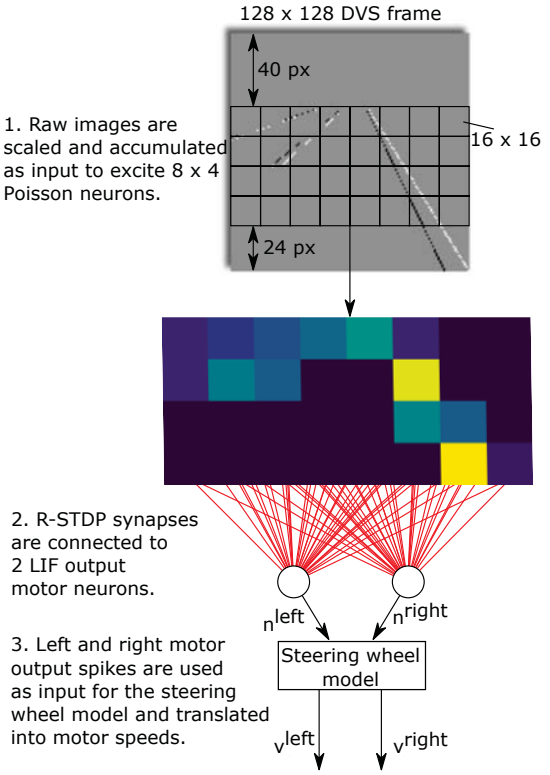


Fig. 5: Network architecture of the R-STDP implementation using DVS frames as input.

A simple learning rule combining models of STDP and a global reward signal was proposed by Izhikevich [22] and Florian [15]. In the R-STDP, the synaptic weight w changes with the reward signal R. The eligibility trace of a synapse can be defined as,

$$\dot{c}(t) = -\frac{c}{\tau_c} + W(\Delta t)\delta(t - s_{pre/post})C_1 \tag{4}$$

where c is an eligibility trace. $s_{pre/post}$ means the time of a pre- or post-synaptic spikes. C_1 is a constant coefficient. τ_c is a time constant of the eligibility trace. Δ is the Dirac delta function.

$$\dot{w}(t) = R(t) \times c(t) \tag{5}$$

where $R(t)$ is the reward signal. More details of R-STDP mechanism can be found in [23, 24].

B. DVS Input Generation

Dynamic vision sensors are biologically inspired vision sensors with a continuous output of independent pixel events and a temporal resolution in the order of microseconds. In order to reduce noise in the images, the DVS camera is set up to only detect the road markings and other deliberately placed objects in the simulation. Intensity changes on the ground for example are ignored. First, in order to reduce the computational complexity of the task, images are reduced to a lower resolution as well. Second, due to the event-based nature of the DVS data, image frames coming from the simulation do not always contain sufficient information for the network to make meaningful decisions. Therefore,

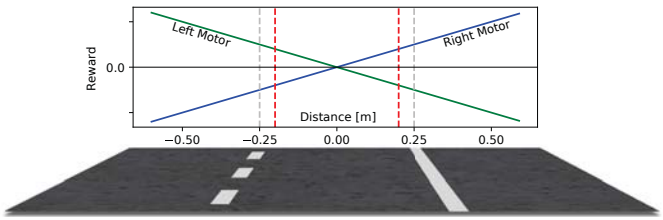


Fig. 6: Reward given by the R-STDP controller: It is defined for each motor individually as a linear function of the lane-center distance scaled by a constant c_r. The lane markings are $0.25\,m$ away from the lane-center. If the robot will go further than $0.2\,m$ from the lane-center, episodes are terminated and the robot will be positioned at its starting position.

the state input was computed by condensing information of several consecutive DVS frames into a single image. As shown in Fig. 4, this is done by dividing the original 128×128 DVS frames into small 4×4 regions and counting every event over ten consecutive frames regardless of the polarity. Furthermore, the image is cropped at the top and bottom resulting in a 32×16 image.

DVS frames are calculated and published every 50ms (with every simulation time-step). Actions are executed every 500ms. Therefore, during one action step, DVS frames are stored in a first-in-first-out (FIFO) queue of length 10. And the last 10 DVS frames are then converted into the final state input.

C. Reward Generation

Instead of dividing the input data and feeding it into two separate networks with static weights as [9], a single SNN based on R-STDP is designed as Fig. 5. The input data is scaled and used for excitation of Poisson neurons, in a single network with $8 \times 4 = 32$ input neurons. Then, the input layer is connected to two LIF output neurons in an "all to all" fashion using R-STDP synapses. The reward signal given at each simulation time step is shown in Fig. 6. It is defined for each motor with opposite signs linearly depending on the robot's distance to the lane-center. When the robot is right from the lane-center and should turn left to get back, connections that lead the right motor neuron to fire are strengthened, connections that lead the left motor neuron to fire are weakened. On the opposite side of the lane-center this process is turned around. Over time, the robot should learn to associate certain input stimuli with left or right turns and act accordingly. These considerations lead to the following rewards for left and right motor neuron connections with d being the distance to the lane-center and c_r a constant scaling the reward:

$$r_{left/right} = -/+(d \cdot c_r) \tag{6}$$

D. Encoding and Decoding

For communicating with robot sensors and motors in SNNs, the sensory information should be encoded into input spikes and the output spikes should be decoded into motor commands. Similar processing procedure for the encoding and decoding can be found in [9] The same model is implemented in this paper as well with only one change. Instead of steering angles, turn speeds are computed and added or subtracted for left and right motor. First, the output spike count $n_t^{left(right)}$ is scaled by the maximum possible output n_{max}:

$$m_t^{left(right)} = \frac{n_t^{left(right)}}{n_{max}} \in [0;1], \text{ with } n_{max} = \frac{T_{sim}}{T_{refrac}}, \tag{7}$$

where 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 c_{turn}, the turn speed is defined as

$$S_t = c_{turn} \cdot a_t, \text{ with } a_t = m_t^{left} - m_t^{right} \in [-1;1]. \tag{8}$$

Furthermore, in order to ensure slower speed in turns, the overall speed is controlled according to

$$V_t = -|a_t| \cdot (v_{max} - v_{min}) + v_{max}, \tag{9}$$

where v_{min} and v_{max} are predefined speed limits. Since controlling a car is generally a continuous process, overall and turn speed (v_t and s_t) were smoothened based on the activities:

$$v_t = c \cdot V_t + (1-c) \cdot v_{t-1}, \tag{10}$$

$$s_t = c \cdot S_t + (1-c) \cdot s_{t-1}, \tag{11}$$

$$\text{with } c = \sqrt{\frac{(m_t^{left})^2 + (m_t^{right})^2}{2}} \tag{12}$$

Finally, the control signals for the left and right motor were computed by

$$v_t^{left} = v_t + s_t \text{ and } v_t^{right} = v_t - s_t. \tag{13}$$

E. Training

In order to train the network successfully, the parameters of the R-STDP controller have to be carefully chosen (See Tab. I in Appendix). First, the training result is closely related to the reward in 6. If the value is too low, the learning will take too much time and it might be difficult to see any progress at all. If it is too high, on the other hand, the learning will get increasingly instable and the robot will not learn anything. Second, the initial network weights are critical for learning as well. In this work, weights are initialized uniformly at a relatively low value of 200. The weights have to be larger than zero, because both motor neurons must be excited from the beginning in order to induce weight changes following the R-STDP learning rule. In the best case the initial weight values are as close as possible to their final values after learning. Therefore, the initial weight value has been set to an estimated mean value of the weights after learning. Furthermore, the weights are clipped to $[0:3000]$ only allowing excitatory synaptic connections.

In Fig. 7 the training progress of the R-STDP controller in the first scenario is shown. Specifically, it shows the termination position of the robot at each trail when it exceeds the

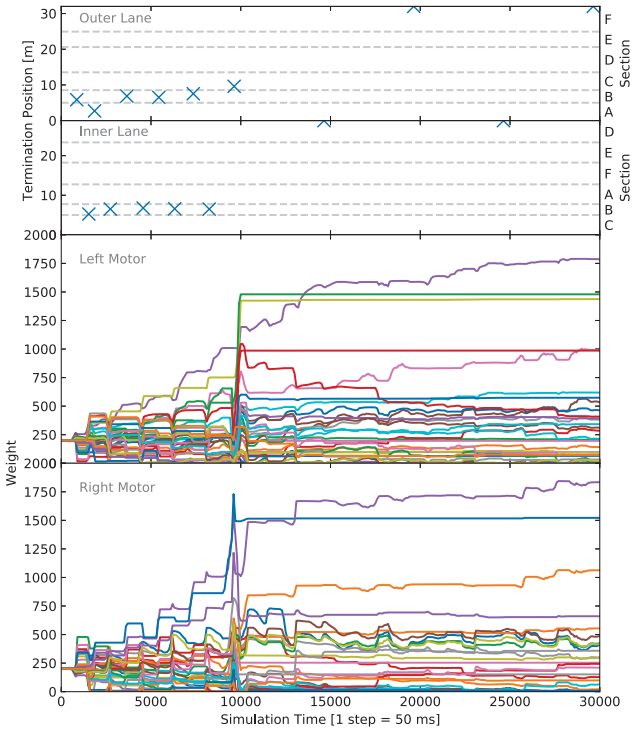


Fig. 7: Scenario 1. Learning progress of the R-STDP controller. The termination position and the network weights are shown over the number of simulation steps ($1\,step = 50\,ms$). During the first 10000 simulation steps, the robot causes resets at each trial in the first turn in both directions (Section B). Afterwards, it has successfully learned how to keep the lane only causing a reset when a complete lap is finished.

lane-center distance of 0.2 m and causes a reset. Moreover, the changes of the synaptic weights are shown over the course of the simulation. A simulation step is equivalent to 50 ms both for the simulation of the SNN as well as the robot simulator itself. In the beginning of the training procedure, the robot will go straight forward, because all connection weights for both motor neurons have been set to the same value. Therefore, during the first 10000 simulation steps, trials are mostly terminated at the first turn in both directions when the robot misses the turn and the lane-center distance exceeds $0.2m$. Each time the robot misses a turn, it will periodically induce high reward values in the beginning changing the synaptic weights. Shortly before step 10000, the robot has learned to take the turn, but it still deviates from the optimal lane-center position. Consequently, the high reward over a longer period of time causes a significant change in the connection values. Evidence for that can also be found by looking at the termination position on the outer lane shortly before step 10000 that lies beyond the first turn. Afterwards, the controller follows the lane in both directions without causing a reset. Episodes are only terminated once the robot has completed a lap. Following both lanes close to the optimal lane-center position means low reward values as well. Therefore, the weight changes after step 10000 are considerably smaller than before. The learned weights after

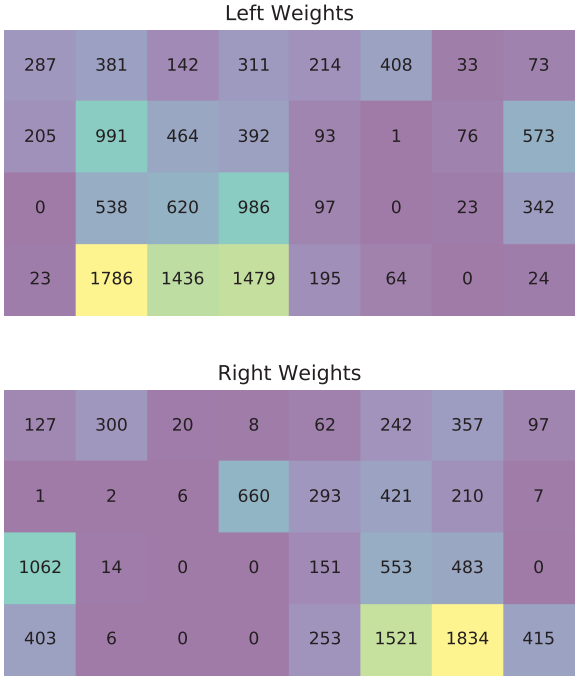


Fig. 8: Scenario 1. Learned connection weights to the left and right motor neuron of the R-STDP controller after 30000 simulation steps.

30000 simulation steps are shown in Fig. 8. Interestingly, the connection weights resemble the theoretically derived weights of the Braitenberg controller with very low values at one half of the image and increasing values from the top corner to the bottom center at the other half of the image. Furthermore, it can be seen that left and right motor neurons seem to be triggered mostly through middle and right road line enclosing the lane.

IV. Discussion

In this section, two more lane-keeping scenarios are also implemented to inspect the practicability of our algorithm as well as the performance comparison with Braitenberg controller [9].

A. Different Task Scenarios

To examine the practicality of the proposed algorithm, another two lane scenarios are implemented (See Fig. 3b and Fig. 3c).

The training progress of the controller in scenario 2 is shown in Fig. 9 and seems very similar to the first scenario, completing the first full lap in less than 5000 simulation steps. The weights of the controller network after 30000 simulation steps are shown in Fig. 10. While the networks weights on the left side from both motor neurons resemble the connection weights learned in scenario 1, it can easily be seen that the weights on the right side have been left unchanged, due to the missing lines in this scenario and the consequential lack of activity during training.

Fig. 11 shows the learning progress during training in scenario 3. First, learning a successful control strategy takes

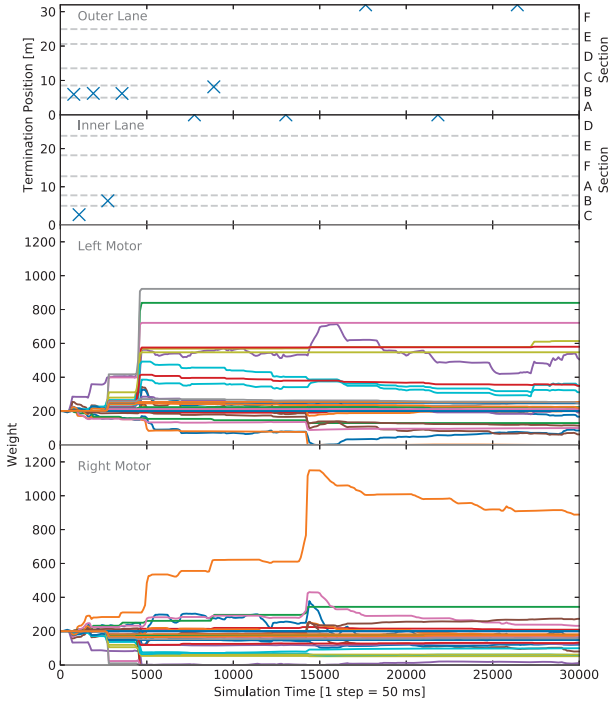


Fig. 9: Scenario 2. The termination position when the robot causes a reset and the network weights are shown over the number of simulation steps.

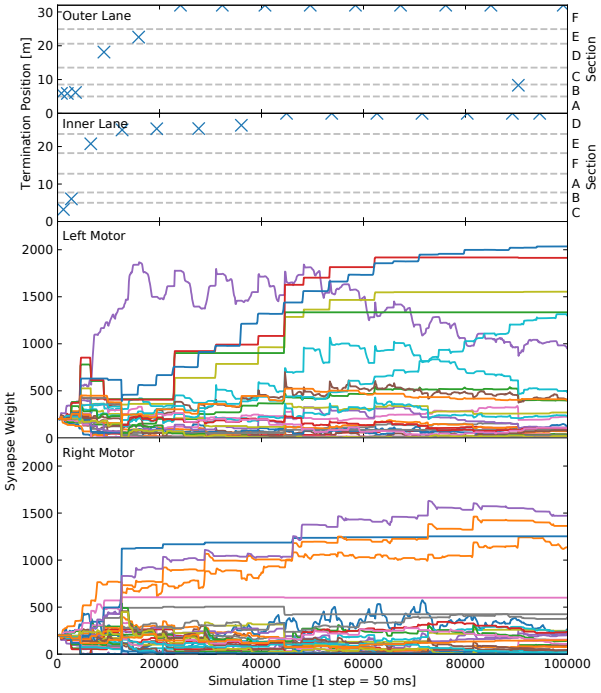


Fig. 11: Scenario 3. After an initial learning phase, the robot is mostly reset in sections B and D until laps are completed on both lanes after approximately 75000 steps.

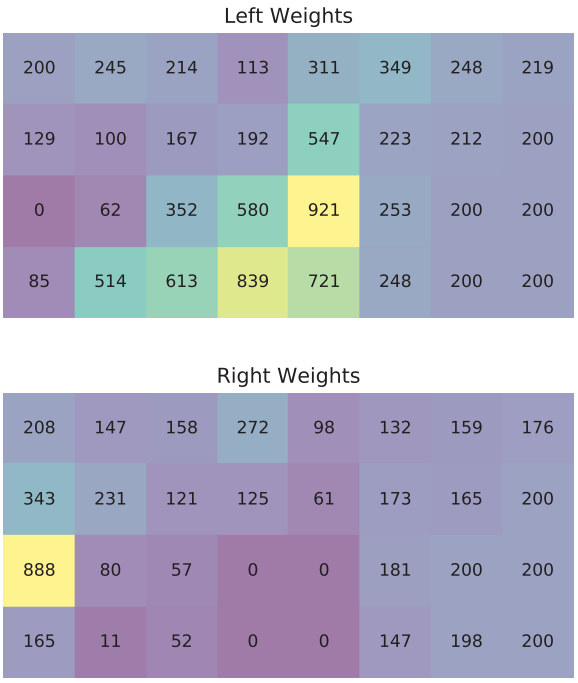


Fig. 10: Scenario 2. Learned connection weights to the left and right motor neuron of the R-STDP controller after 30000 simulation steps.

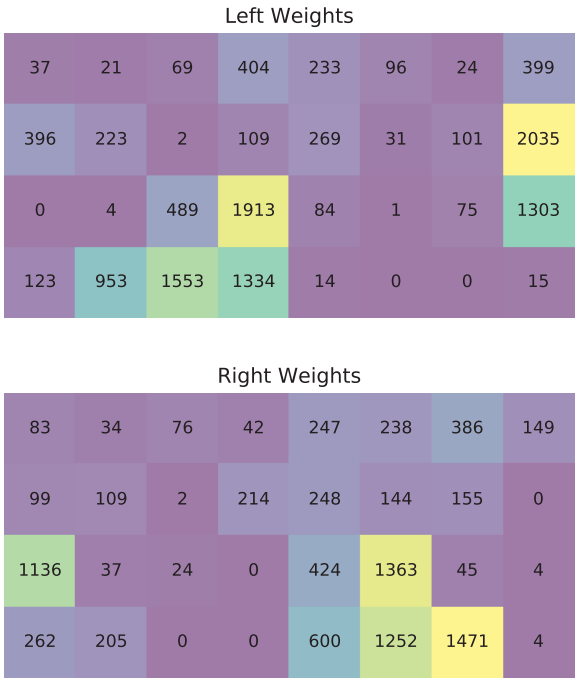


Fig. 12: Scenario 3. Learned connection weights to the left and right motor neuron of the R-STDP controller after 100000 simulation steps.

considerably more time than in the first two scenarios. The obvious explanation for this is that scenario 3 incorporates two different road patterns making the environment more complicated. Therefore, the controller has to distinguish between a higher number of different situations as well as slowing down the learning procedure. Moreover, due to the simple fact that the robot does not encounter certain situations until it has learned how to get there, it will

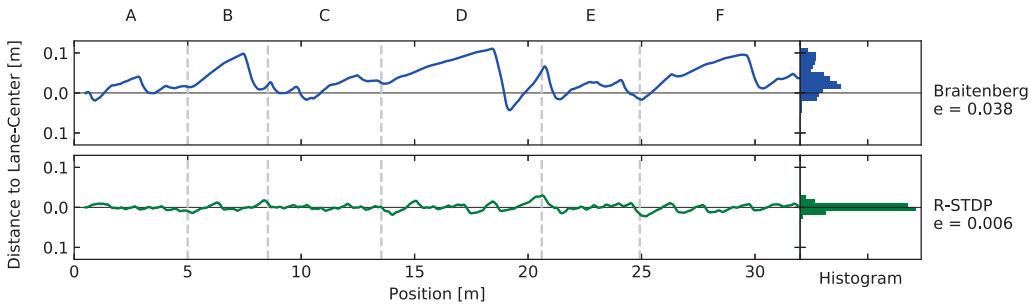


Fig. 13: Comparison of the different controllers on the outer lane of the simple lane keeping scenario. The deviation from the lane-center is shown over the robot position projected to the lane-center. Positive lane-center distances correspond to deviations to the right side, negative distances to the left side. Course sections are marked by vertical dashed lines (A=straight, B=left, C=straight, D=left, E=right, F=left). On the right side, error distributions for all controllers as well as mean errors e (mean distance to the lane-center) are shown.

only start learning a generalized control strategy that works for both lanes towards the end. In the left motor plot it can be seen that some weights might even be increased in the beginning and decreased again afterwards. After an initial learning phase until approximately step 20000, the controller is mostly reset in section B (outer lane) and D (inner lane). When the weights have adapted sufficiently after approximately 75000 steps, the robot finished the laps on both lanes. In Fig. 12, the learned weights after 100000 steps are shown. In comparison to the first scenario the weight patterns seem very similar, which makes sense considering the fact that the road pattern in scenario 1 is the combination of both road patterns in scenario 3.

B. Performance Comparison

To verify the superiority of the proposed algorithm, the performance is compared to the Braitenberg controller in [9]. To get comparable performance metrics for both controller, they are evaluated completing one lap on the outer lane in the first scenario. Fig. 13 shows the deviation of the robot from the lane-center over the projected course position during the performed lap. Moreover, the course is divided into the six sections as shown in Fig. 3a. The robot path is represented as a projection to the lane-center line, which allows for a numerical analysis of the controllers performance. Specifically, the error distribution (distance to the lane-center) can be shown in the form of a histogram as well as the mean error for each controller.

First, the Braitenberg controller is evaluated performing the same lap. While the controller successfully finishes the course, it can be seen clearly that it strongly tends to the right side of the lane (*left*: negative, *right*: positive, See Fig. 5), which can be explained by the vision field of the robot. In the right half of the DVS images, the robot usually only sees the right solid line. In the left half, however, the robot sees the left solid line as well as the dashed middle line of the road, leading to a higher number of detected events and a higher activity of the left motor neuron eventually. This will shift the robot to the right until it reaches a balance in the activity of the motor neurons. Even in the right turn (section E) the robot is mostly left from the lane-center. In left turns the distance to the lane-center grows until a point is reached where previously unstimulated neurons with high weights are now excited. These will push the right motor neuron activity leading to a movement correction back to the center. This can be seen in all 3 left turns (sections B, D and F).

Of both controllers, the R-STDP controller shows the better performance in this task with comparatively very small deviations from the lane-center. This gets even clearer when looking at the performance histogram and the mean error that is almost an order of magnitude lower than the ones before. First, one explanation for this behavior can be found in the very nature of SNNs that allow for high frequency decision making without the need of splitting time into discrete steps. Second, the R-STDP training algorithm and the related reward are to a great extent tailored to this specific problem. Basically, the R-STDP reward can be interpreted as a pre-defined value function with a global maximum that the algorithm will seek. Therefore, the R-STDP training algorithm leaves out the state evaluation step that is typical for every classical reinforcement learning algorithm.

V. Conclusion and Outlook

R-STDP learning rule, by combining the advantages of the reinforcement learning and STDP mechanism, offers a promising solution to train SNNs. However, it lacks of practical robotic implementations since its complexities in constructing and training a SNN. To bridge this gap, we have trained a SNN controller based on R-STDP and implemented it in lane-keeping tasks for a Pioneer robot. First, with the advantages of DVS for data acquisition, our algorithm tends to be fast and robust from illumination conditions. Further, this algorithm is capable of learning to follow different road patterns, even if they are changing within a single scenario. Finally, comparing to the static SNN controller, the proposed algorithm exhibits better performance in terms of the deviation of the robot from the center line.

For future work, the R-STDP controller is build as first step towards more sophisticated algorithms with real reinforcement learning capabilities. Currently research has not

incorporated reward prediction errors yet, even though this phenomenon was observed in the brain. Therefore, such networks based on R-STDP should be also implemented using deep architectures in the future.

APPENDIX

All the simulation parameters are listed in Tab. I.

TABLE I: Simulation parameters specification

Steering model	max. speed	$v_{max} = 1.5 m/s$
	min. speed	$v_{min} = 1.0 m/s$
	turn constant	$c_{turn} = 0.5$
	max spikes during a simulation step	$n_{max} = 15$
Poisson neurons	max. firing rate	300 Hz
	number of DVS events for max. firing rate	$n = 15$
SNN simulation	simulation time	50 ms
	time resolution	0.1 ms
LIF	NEST model	iaf_psc_alpha
	Resting membrane potential	$E_L = -70.0 mV$
	Capacity of the membrane	$C_m = 250.0 pF$
	Membrane time constant	$\tau_m = 10.0 ms$
	Time constant of postsynaptic excitatory currents	$\tau_{syn,ex} = 2.0 ms$
	Time constant of postsynaptic inhibitory currents	$\tau_{syn,in} = 2.0 ms$
	Duration of refractory period	$t_{ref} = 2.0 ms$
	Reset membrane potential	$V_{reset} = -70.0 mV$
	Spike threshold	$V_{th} = -55.0 mV$
	Constant input current	$I_e = 0.0 pA$
R-STDP synapse	NEST model	dopamine_synapse
	Amplitude of weight change for facilitation	$A_+ = 1.0$
	Amplitude of weight change for depression	$A_- = 1.0$
	STDP time constant for facilitation	$\tau_+ = 20.0 ms$
	Time constant of eligibility trace	$\tau_c = 1000.0 ms$
	Time constant of dopaminergic trace	$\tau_n = 200.0 ms$
	Minimal synaptic weight	0.0
	Maximal synaptic weight	3000.0
	Initial synaptic weight	200.0
	Reward constant	$c_r = 0.01$

ACKNOWLEDGEMENT

The research leading to these results has received funding from grant agreement No. 67000-42070002, the European Union Research and Innovation Programme Horizon 2020 (H2020/2014-2020) under grant agreement No. 720270 (The Human Brain Project, HBP), and the Chinese Scholarship Council.

REFERENCES

[1] D. Drubach, *The brain explained*. Prentice Hall, 2000.

[2] R. Brette, "Philosophy of the spike: Rate-based vs. spike-based theories of the brain," *Frontiers in Systems Neuroscience*, vol. 9, p. 151, 2015. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnsys.2015.00151

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

[4] 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

[5] G. Indiveri, "Neuromorphic analog vlsi sensor for visual tracking: Circuits and application examples," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 11, pp. 1337–1347, 1999.

[6] M. A. Lewis, R. Etienne-Cummings, A. H. Cohen, and M. Hartmann, "Toward biomorphic control using custom avlsi cpg chips," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, 2000, pp. 494–500 vol.1.

[7] A. Ambrosano, L. Vannucci, U. Albanese, M. Kirtay, E. Falotico, P. Martínez-Cañada, G. Hinkel, J. Kaiser, S. Ulbrich, P. Levi, C. Morillas, A. Knoll, M.-O. Gewaltig, and C. Laschi, *Retina Color-Opponency Based Pursuit Implemented Through Spiking Neural Networks in the Neurorobotics Platform*. Cham: Springer International Publishing, 2016, pp. 16–27. [Online]. Available: https://doi.org/10.1007/978-3-319-42417-0_2

[8] X. Wang, Z. G. Hou, M. Tan, Y. Wang, and L. Hu, "The wall-following controller for the mobile robot using spiking neurons," in *2009 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 1, Nov 2009, pp. 194–199.

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

[10] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[11] 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, pp. 655–666, 2008.

[12] X. Wang, Z.-G. Hou, F. Lv, M. Tan, and Y. Wang, "Mobile robots modular navigation controller using spiking neural networks," *Neurocomputing*, vol. 134, pp. 230–238, 2014.

[13] Y. Bengio, T. Mesnard, A. Fischer, S. Zhang, and Y. Wu, "Stdp-compatible approximation of backpropagation in an energy-based model," *Neural computation*, 2017.

[14] 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.

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

[16] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb 2008.

[17] E. Nichols, L. J. McDaid, and N. Siddique, "Biologically inspired snn for robot control," *IEEE transactions on cybernetics*, vol. 43, no. 1, pp. 115–128, 2013.

[18] A. MobileRobots, "Pioneer p3-dx," *Website. http://www. mobilerobots. c om/ResearchRobots/PioneerP3DX. aspx*, 2012.

[19] N. Caporale and Y. Dan, "Spike timing–dependent plasticity: a hebbian learning rule," *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, 2008.

[20] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps," *Science*, vol. 275, no. 5297, pp. 213–215, 1997.

[21] G. Bi and M. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, 1998.

[22] 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.

[23] W. Potjans, A. Morrison, and M. Diesmann, "Enabling functional neural circuit simulations with distributed computing of neuromodulated plasticity," *Spike-timing dependent plasticity*, p. 357, 2010.

[24] N. Frmaux and W. Gerstner, "Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules," *Frontiers in Neural Circuits*, vol. 9, p. 85, 2016. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fncir.2015.00085