Biologically Inspired Reinforcement Learning for Mobile Robot Collision Avoidance

Myung Seok Shim and Peng Li
Department of Electrical and Computer Engineering
Texas A&M University, College Station, Texas 77843, USA
Emails: {mrshim1101, pli}@tamu.edu

***Abstract*—Collision avoidance is a key technology enabling applications such as autonomous vehicles and robots. Various reinforcement learning techniques such as the popular Q-learning algorithms have emerged as a promising solution for collision avoidance in robotics. While spiking neural networks (SNNs), the third generation model of neural networks, have gained increased interest due to their closer resemblance to biological neural circuits in the brain, the application of SNNs to mobile robot navigation has not been well studied. Under the context of reinforcement learning, this paper aims to investigate the potential of biologically-motivated spiking neural networks for goal-directed collision avoidance in reasonably complex environments. Unlike the existing additive reward-modulated spike-timing dependent plasticity learning rule (A-RM-STDP), for the first time, we explore a new multiplicative RM-STDP scheme (M-RM-STDP) for the targeted application. Furthermore, we propose a more biologically plausible feed-forward spiking neural network architecture with fine-grained global rewards. Finally, by combining the above two techniques we demonstrate a further improved solution to collision avoidance. Our proposed approaches not only completely outperform Q-learning for cases where Q-learning can hardly reach the target without collision, but also significantly outperform a baseline SNN with A-RM-STDP in terms of both success rate and the quality of navigation trajectories.**

I. Introduction

Collision avoidance is a key enabling technology and is essential to autonomous systems such as mobile robots and self-driving cars. Collision avoidance systems are currently under intensive development in the auto industry [1]. In mobile robotic applications, an autonomous agent typically relies on cameras, lasers, ultrasonic sensors, and certain forms of learning capabilities to avoid collisions in a given environment. Towards this end, various types of reinforcement learning techniques such as adaptive dynamic programming and temporal difference are studied [2], [3]. Also, collision avoidance techniques based on these reinforcement learning approaches have been studied in the literature [4]–[9].

Among these, Q-learning [3], a popular method of reinforcement learning, has been adopted for collision avoidance where an optimal state-action policy is computed based on the Markov Decision Process (MDP). [5] employs Q-learning to train multiple robots to achieve a certain shaped formation and move in the formation while avoiding obstacles. [4] uses an artificial neural network to more efficiently represent the Q-values as part of Q-learning for robot collision avoidance without a target. The considered actions are discrete and correspond to moving and rotating in different directions. It is shown through simulation that the Q-learning algorithm converges in about 500 epochs for environments with a few obstacles [4]. However, the movement of the robot does not look natural due to discrete control actions. In addition, a simplistic reward function which depends only on the actions taken and occurrences of collisions is used.

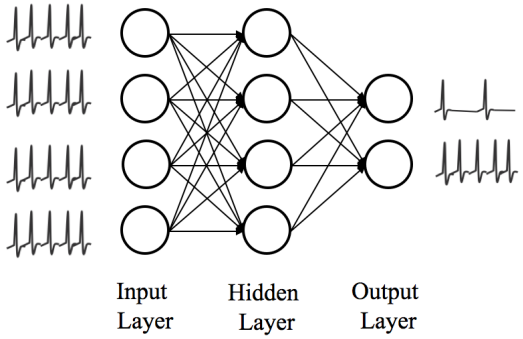

Fig. 1. A feed-forward spiking neural network.

In recent years, spiking neural networks (SNNs), the third generation model of neural networks, have gained increased interest due to their closer resemblance to neural circuits in biological brains. As an example, Fig. 1 shows a feed-forward spiking neural network.

Ideas on exploring reinforcement learning under the context of SNNs, particularly the concept of Reward-Modulated STDP (RM-STDP), have been suggested by the neuroscience community [10]–[12]. To this end, an SNN may learn an optimal policy for decision making under a properly designed global reward signal with eligibility trace for delayed rewards. Note that RM-STDP is modulated by the reward signal and is different from the standard Spike-Timing-Dependent Plasticity (STDP), which is a form of unsupervised Hebbian learning.

While SNN based reinforcement learning and similar techniques are still a relatively new territory of research, there have been some initial attempts towards applying such techniques for robot collision avoidance. [13] presents a SNN-based target tracking robot controller, where a charge coupled device(CCD) camera is used to detect the target and 16 ultrasonic sensors are used to detect obstacles around the robot. The camera inputs are transformed into spike trains and drive directly the output neurons which control the motors. On the other hand, ultrasonic sensor inputs are projected to the hidden

layer to learn collision avoidance, and the outputs of the hidden layers are fed to the output layer. The overall learning approach is not based on reinforcement learning where an unsupervised spike-based Hebbian learning mechanism is adopted to update synaptic weights. In [9], a relatively simple conditionally reward-modulated Hebbian plasticity mechanism is used for learning simple goal-directed behaviors in open environments without any obstacle. [8] employs short-term plasticity and long-term plastiticy realized using the temporal difference (TD) rule for wall following. [14] uses standard STDP for collision avoidance and target approaching in simple environments.

While the application of spiking neural networks has not been well studied for mobile robot navigation, the key objective of this paper is to investigate the potential of biologically-motivated spiking neural networks for goal-directed collision avoidance in reasonably complex environments under the context of reinforcement learning. Our main new contributions are:

- In addition to the existing standard additive RM-STDP scheme (A-RM-STDP), for the first time, we explore a multiplicative RM-STDP scheme (M-RM-STDP) under the context of reinforcement learning and demonstrate its potential.
- We explore a more biologically plausible feed-forward SNN architecture where in the hidden layer, both excitatory and inhibitory neurons are employed. We further develop a fine-grained reward scheme for the new architecture.
- Finally, we demonstrate that by combining the above two techniques leading to a further optimized solution with improved success rates and navigation trajectories.

To demonstrate the proposed ideas, we design several feed-forward SNNs with one input, one hidden and one output layer. Our proposed approaches completely outperform Q-learning in terms of both the success rate and the quality of trajectories to the target.

With respect to our first contribution, multiplicative reward-modulated STDP schemes have not been explored for reinforcement learning. To this end, for the first time, we present a multiplicative reward-modulated STDP scheme (M-RM-STDP). We demonstrate that for the application of mobile robot navigation, our M-RM-STDP outperforms the more conventional additive scheme, i.e. A-RM-STDP, significantly, for example by increasing the success rate by 40% in the testing phase of robot navigation. In addition, M-RM-STDP reduces the number of steering movements taken for reaching the target by 10%.

Our second contribution is motivated by the fact that excitatory and inhibitory mechanisms produce rich behaviors in biological brains. For this, we introduce both excitatory and inhibitory neurons not only in the input layer but also in the hidden layer to improve learning performance. Furthermore, we propose optimized reward functions which specify the amount of reward as a function of the distances to nearby obstacles, the distance to the target, and the angle between the moving direction of the robot and the vector that points to the target from the robot. Very importantly, instead of utilizing a single global reward for all synapses as typically done in the prior work, we take a finer-grained approach that is based on the fact that the inclusion of both exictatory and inhibitory neurons in the first two layers produces feed-forward paths that have rather different effects on each output neuron. As a result, the net effects of two synapses on the same output neuron can be opposite of each other, with one being excitatory and the other inhibitory, depending the nature of the signal paths which these synapses are on. Recognizing this disparity, we conduct additional sign adjustment of the reward for different synapses in the network, which significantly improves performance.

Based on the first two contributions, our combined approach, i.e. the third contribution, produces the best results among studied approaches. It significantly outperforms the standard RM-STDP scheme, for example by increasing the success rate in the testing phase by up to 45% while reducing the number of steering movements taken to get to the target.

II. Background

We briefly discuss the leaky integrate-and-fire (LIF) spiking neuron model, STDP, and additive reward-modulated STDP.

A. The Leaky Integrate-and-Fire Neuron Model

A number of spiking neural models such as the (leaky) integrate-and-fire, Hodgkin-Huxley [15], and Izhikevich [16] models have been adopted for modeling SNNs. In this work, the leaky integrate-and-fire(LIF) model is utilized due to its simplicity:

$$\tau_m \frac{dv(t)}{dt} = -(v(t) - v_{rest}) + RI(t), \tag{1}$$

where $v(t)$ is the membrane potential, τ_m the membrane time constant, v_{rest} the resting potential, R the membrane resistance, and $I(t)$ the synaptic input current. Under some injected $I(t)$, the membrane potential may go beyond the threshold voltage starting from which an action potential, or a spike, would be generated [17].

B. Spike-Timing-Dependent Plasticity

STDP is an unsupervised Hebbian learning mechanism, which adjusts a synaptic weight based upon the spike timing difference between the corresponding pre-synaptic and post-synaptic spikes [18]. The weight w_{ij} is strengthened if the pre-synaptic neuron fires before the post-synaptic neuron, otherwise it is weakened. The temporal difference between the firing times of each pair of the pre-synaptic and post-synaptic spikes $\Delta t = t_{post} - t_{pre}$ determines the amount of weight change:

$$\begin{aligned} \Delta w_+ &= A_+ \cdot e^{-\frac{\Delta t}{\tau_+}} \quad if \quad \Delta t > 0 \\ \Delta w_- &= A_- \cdot e^{-\frac{\Delta t}{\tau_-}} \quad if \quad \Delta t < 0, \end{aligned} \tag{2}$$

where Δw_+ and Δw_- are the weight modifications induced by long-term potentiation (LTP)and long-term depression (LTD),

respectively, A_+ and A_- are some positive and negative constant parameter and determine the strength of LTP and LTD, respectively, and τ_+ and τ_- set the temporal windows over which STDP is active. A typical STDP curve is shown in Fig. 2.

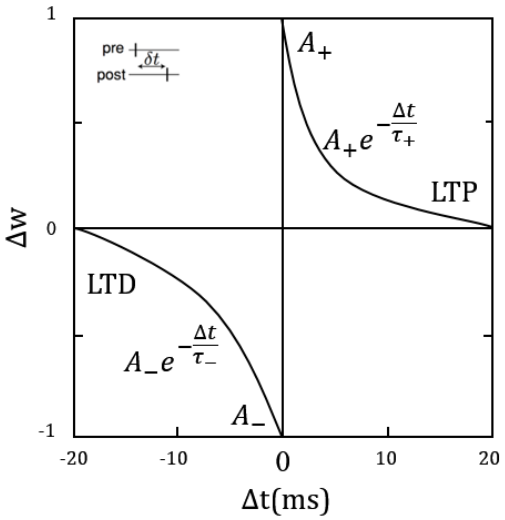

Fig. 2. A typical STDP curve.

C. Additive Reward-Modulated STDP

Additive reward-modulated STDP (A-RM-STDP) is an implementation of reinforcement learning mechanism, which updates the synaptic efficacy in an additive manner. While STDP operates based upon the correlation between the spike timings of the pre- and postsynaptic neurons, a reward is introduced to modulate STDP in A-RM-STDP. If the reward is positive, the corresponding synapse is potentiated. Otherwise, it is depressed.

For a synapse projecting from the j-th neuron to the i-th neuron, A-RM-STDP may be realized according to [10]:

$$\frac{dw_{ij}(t)}{dt} = \gamma r(t) z_{ij}(t), \tag{3}$$

where γ is the learning rate, $r(t)$ the reward signal, and $z_{ij}(t)$ the eligibility trace which decays the amount of weight update over time. The eligibility trace can be defined as:

$$\tau_z \frac{dz_{ij}(t)}{dt} = -z_{ij}(t) + \xi_{ij}(t), \tag{4}$$

where additional dynamic variables ${P_{ij}}^+$, ${P_{ij}}^-$ and ξ_{ij} are utilized to track the effect of pre-synaptic and post-synaptic spikes [10]:

$$\xi_{ij}(t) = {P_{ij}}^+ \Phi_i(t) + {P_{ij}}^- \Phi_j(t), \tag{5}$$

$$\frac{d{P_{ij}}^+(t)}{dt} = -\frac{{P_{ij}}^+(t)}{\tau_+} + A_+ \Phi_j(t), \tag{6}$$

$$\frac{d{P_{ij}}^-(t)}{dt} = -\frac{{P_{ij}}^-(t)}{\tau_-} + A_- \Phi_i(t), \tag{7}$$

where Φ_i is the Dirac delta function which is non-zero only at times t when the neuron i fires, and A_+ and A_- are certain positive and negative constant, respectively. A typical RM-STDP characteristics is shown in Fig. 3.

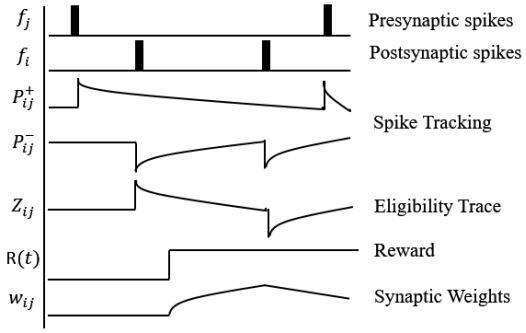

Fig. 3. A typical additive RM-STDP characteristics.

III. Reinforcement Learning with the Proposed Multiplicative RM-STDP

While multiplicative reward-modulated STDP has not been studied for reinforcement learning, for the first time we present a multiplicative reward-modulated STDP scheme (M-RM-STDP) which shows good performance for mobile robot navigation. We then present an optimized reward function which specifies the amount of reward as a function of the distance to obstacles, the distance to the target, and the angle between the moving direction of the robot and the vector that points to the target from the robot.

A. Multiplicative Reward-Modulated STDP

In the past, only additive reward-modulated STDP has been considered for reinforcement learning under the context of spiking neural networks. In this paper, in contrast to (3), we explore a new multiplicative scheme as follows:

$$\frac{dw_{ij}(t)}{dt} = \gamma w_{ij}(t) r(t) z_{ij}(t). \tag{8}$$

As can been seen, in this new M-RM-STDP scheme, the synaptic weight is updated based on the product of four components: the current weight, learning rate, reward, and eligibility trace. As a result, the instantaneous rate of weight change is proportional to the current weight value $w_{ij}(t)$.

Under different contexts, earlier study on multiplicative STDP demonstrates that it may behave differently from additive STDP, for example, by producing stable unimodal distributions of synaptic weights, which makes synapses less sensitive to input perturbations [19]. Under the context of SNN-based reinforcement learning for mobile robot navigation, we have experimentally observed that in addition to possible performance boost, the maximum value of weight change resulted from the proposed M-RM-STDP increases rather noticeably, for example, by five times in some cases over A-RM-STDP. As a result, it is also observed that the multiplicative nature of the proposed RM-STDP scheme leads to faster learning.

B. Proposed Reward Functions

In this work, we employ feed-forward spiking neural networks with an input layer, hidden layer, and output layer. The output layer consists of two output neurons controlling the left and right motors of the robot, i.e. the output neuron that fires with a higher frequency makes the robot turn based on the corresponding motor. More details about the network setting are provided in Section V.

Reward functions play an important role for reinforcement learning as they provide critical feedback from the environment to the agent. In conjunction with the existing A-RM-STDP and proposed M-RM-STDP schemes, we make use of reward functions optimized for the targeted application. Unlike the simple reward function used in [4], which only depends on actions and occurrences of collision, the proposed reward functions takes the distance to obstacles, the distance to the target, and the angle between the moving direction of the robot and the vector that points to the target from the robot into consideration. In this section, we describe how rewards are computed. The specific ways in which the rewards are applied to the network will be discussed in the next section.

1) Rewards for Collisions and Arrival: During each training trial, when the robot collides with an obstacle, the entire network gets a negative reward of R_{col}. Otherwise, if the robot arrives at the target, the network gets a positive reward of R_{arr}. In our experiments, we set: $R_{col} = -2.0$ and $R_{arr} = 2.0$. In both cases, the training trial ends after the application of the reward.

2) Rewards for Collision Avoidance: In the absence of collisions and arrival at the target, we compute the following two rewards to promote collision avoidance when the robot gets close to obstacles, which may be detected, for example, by ultrasonic sensors:

$$\begin{aligned} r_1(d) &= +\frac{d_{safe} - d}{d_{safe}} + k, \quad for \quad d < d_{safe} \\ r_2(d) &= -\frac{d_{safe} - d}{d_{safe}} - k, \quad for \quad d < d_{safe} \end{aligned} \tag{9}$$

where d is the distance to the nearest obstacle. In our simulation environment, we set d_{safe} to 100 pixels: $d_{safe} = 100$, and constant k to 0.3 for A-RM-STDP and 1.3 for M-RM-STDP, respectively. The reward values are earned experimentally for avoiding deadlock situations.

More specifically, if the nearest obstacle is within d_{safe} from the robot on the right, the negative reward r_2 is applied to the subset of synapses that influence the firing of the left motor (output) neuron while the positive reward of r_1 is applied to the remaining synapses of the network, which influence the right motor (output) neuron. In this case, there is a tendency for the right motor to rotate faster than the left motor, steering the robot to the left. We swap the roles of r_1 and r_2 if the nearest obstacle is within d_{safe} from the robot on the left.

3) Reward in the Vicinity of the Target: After conditionally applying the rewards specified in (9), we further check if the robot is in the vicinity of the target, i.e. if the distance d between the target and robot is less than a specified threshold d_{tar}. If so, the following additional award is computed:

$$r_3(d) = \frac{1}{d_{tar}} (d_{tar} - d) + 1.0, \quad for \quad d < d_{tar}, \tag{10}$$

where d_{tar} is set to 200 pixels in our experiments.

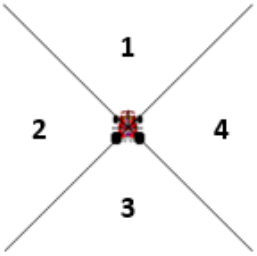

Fig. 4. Four intervals for the angle between the moving direction of the robot and the vector that points to the target from the robot.

We further consider the angle between the moving direction of the robot and the vector that points to the target from the robot. As shown in Fig. 4, we split the environment into four regions around the robot. If the target falls into the regions "2" and "4", the reward r_3 is then applied to the synapses influencing the left and right motor neuron, respectively. No reward is applied when the target falls in the regions "1" and "3".

IV. Proposed Feed-forward SNNs and Fine-grained Rewards

A. Feed-forward SNNs

Cortical circuits in biological brains operate based upon both excitatory and inhibitory mechanisms. A proper balancing between excitation and inhibition in feed-forward networks has been shown to be beneficial [20]. While earlier works have employed both inhibitory and excitatory neurons only in the input layer [10], [21], we extend by doing the same for the hidden layer as well as shown in Fig. 5. Here, the synapses from each excitatory (inhibitory) neuron are considered to be excitatory (inhibitory) in nature.

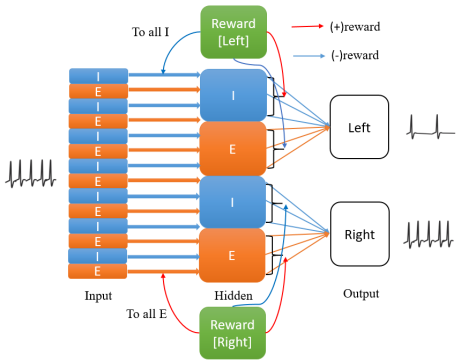

Fig. 5. The proposed spiking neural network with fine-grained rewards.

While the above approach improves learning performance as demonstrated by our experimental results, such improvements

can only be achieved if and only if each reward signal is properly applied to the network. In particular, the inclusion of both excitatory and inhibitory neurons in the network nevertheless introduces certain complications in the application of rewards. Our experiments have shown that treating each reward as a "global" signal and applying it uni-formally to all synapses in the network, as what is typically done in the literature, can lead to very poor learning performance. To address this problem, we propose a fine-grained approach for applying a reward to the network as discussed next.

B. Fine-Grained Rewards

We first recognize that there exist four types of feed-forward paths from the input layer, to the hidden layer, and finally to the output layer in the network of Fig. 5: Inhibitory-Inhibitory (I-I), Inhibitory-Excitatory (I-E), Excitatory-Inhibitory (E-I), and Excitatory-Excitatory (E-E), where the first designation specifies the synapse type from the input to the hidden layer and the second the synapse type from the hidden to the output layer on the path.

Recall that Section III-B describes several different types of reward. Each reward may be applied to all synapses or just a subset of them in the network. Once obtaining the value of a reward, we do not immediately apply the reward to the targeted synapses. Instead, additional sign adjustment may be performed to properly deal with each of the four types of feed-forward signal paths.

To explain our idea, let us consider the following illustrative example for which a reward value of r is computed based on one of the reward functions described in Section III-B. Let us further assume that this reward is intended for the synapses influencing the right motor (output) neuron to incentivize the right motor to rotate faster than the left motor to make the robot turn left. In this case, we consider all four different type paths ending at the right motor neuron.

To achieve our goal, we potentiate or depress each synapse on a given signal path as follows (Fig. 5):

- I-I: potentiate the first with a reward of r; depress the second with a reward of $-r$;
- I-E: depress the first with a reward of $-r$; potentiate the second with a reward of r;
- E-I: depress the first with a reward of $-r$; depress the second with a reward of $-r$;
- E-E: potentiate the first with a reward of r; potentiate the second with a reward of r;

Note that here depressing an inhibitory synapse with a negative reward means reducing the absolute value of the synaptic weight. The basic idea behind the above fine-grained reward approach is to consider the excitatory or inhibitory nature of each synapse with respect to the targeted output neuron. For example, for the E-I type signal paths, while the first synapse is excitatory, its effect on the right motor neuron is in fact *inhibitory*. As a result, we depress this synapse with a negative reward of $-r$.

V. Experimental Settings

To demonstrate the proposed reinforcement learning approach, we consider the problem of autonomous mobile robot navigation towards a fixed target in environments with stationary obstacles. As shown in Fig. 6, we assume that the targeted robot has five ultrasonic sensors to measure distances to nearby obstacles. In addition, we also assume that the distance from the current location to the target and the angle between the moving direction of the robot and the vector that points to the target from the robot are also available to the robot through a "distance" and "angle" sensor, respectively. Note that the robot is modeled simplistically without internal delay and motor control dynamics. The adopted simulation environment is based on Pygame 1.9.2, a game library in Python, and Brian 1.41 [22], an SNN simulator in Python.

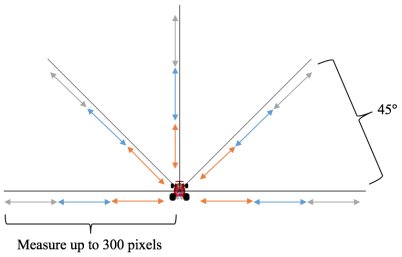

Fig. 6. The adopted car model.

A. Spiking Neural Networks

We employ feed-forward spiking neural networks of three layers: an input, hidden, and output layer. The input layer is composed of seven groups of Poisson neurons for generating spike trains encoding the inputs from the five ultrasonic sensors, distance sensor and angle sensor. Each group has 15 excitatory and 15 inhibitory neurons.

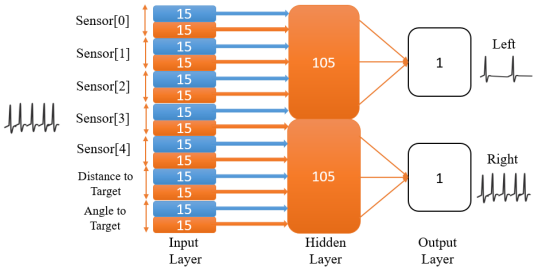

Fig. 7. The reference SNN with only excitatory neurons in the hidden layer.

To demonstrate the SNN architecture proposed in Section IV-A, we consider two compositions for the hidden layer: 210 excitatory neurons (Fig. 7) vs. 104 excitatory and 104 inhibitory neurons (Fig. 8), with the former being a reference and the latter representing the proposed architecture. In both networks, the input layer is fully connected to the hidden layer. The hidden layer is split into two equal halves, with each projecting to one of the two output neurons with fully connectivity. The two output neurons control the left and right motors of the robot and can steer the robot to right and left, respectively.

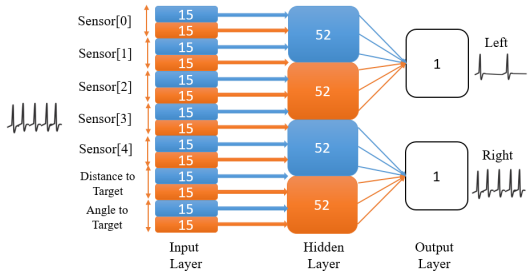

Fig. 8. The proposed SNN with both excitatory and inhibitory-neurons in the hidden layer.

B. State Space and Discretization

The sensory inputs from the five ultrasonic (measuring the distances to nearby obstacles), one distance (measuring the distance to the target) and one angle sensor form the seven-dimensional state of the robot. Each of the five ultrasonic sensors can measure up to 300 pixels. For the reference SNN, every ultrasonic sensor input is discretized into three intervals with each encoded using a different firing frequency: 0 to 99 pixels (0 Hz), 100 to 199 pixels (20 Hz), and 200 to 300 pixels (40 Hz). The encoding of the ultrasonic sensor inputs is done somewhat differently for the proposed SNN: 0 to 99 pixels (0 Hz), 100 to 199 pixels (50 Hz), and 200 to 300 pixels (100 Hz). Each frequency value is used to set the firing frequency of the Poisson neurons in the corresponding input neuron group.

The distance sensor can measure up to 800 pixels for the distance to the target. Its input range (0 to 800 pixels) is divided evenly into four intervals. For the reference SNN, the encoded Poisson firing frequency is 10 Hz, 20 Hz, 30 Hz and 40 Hz, respectively for these intervals. For the proposed SNN, the encoded frequencies are 25 Hz, 50 Hz, 75 Hz, and 100 Hz.

Finally, the input from the angle sensor is discretized into four intervals: north, south, west and east direction, as shown in Fig. 4. The frequency encoding schemes used for the distance sensor are adopted for the angle sensor.

C. Motor Control

The speed of the robot is fixed as 15 pixels/sec such that the robot always moves forward. At each simulation time step, the instantaneous firing rates of the two output (motor) neurons are used to determine the steering angle, that is, the steering angle θ is set to $\theta = freq_l - freq_r$, where $freq_l$ and $freq_r$ are the firing frequencies of the left and right motor neuron, respectively. As such, θ is continuous-valued and the robot is steered to the left if θ is negative. Note that, in Q-learning, ϵ-greedy with ϵ set to 0.3 is used for selecting from three actions: move forward, turn left by 30° and turn right by 30°.

D. STDP parameter settings

The parameter settings of the A-RM-STDP and M-RM-STDP schemes are given in Table I. The initial weights of excitatory synapses are set randomly between 0 - W_{max} mV while those of inhibitory synapses are set randomly between W_{min} - 0 mV. For the proposed network architecture in Fig. 8, the maximum synaptic weight is set to $5mV$ for the half of the synapses between input and hidden layer, and $2.5mV$ for the remaining half.

TABLE I
PARAMETER SETTINGS OF A-RM-STDP AND M-RM-STDP.

Parameter	Value
A_+	$1mV$
A_-	$-1mV$
τ_+	$20ms$
τ_-	$20ms$
W_{max}	$5mV$
W_{min}	$-5mV$

VI. EXPERIMENTAL RESULTS

By using the setups described previously, we compare four spiking neural networks and Q-learning as summarized in Table II. Two spiking neural networks have both excitatory and inhibitory neurons only in the input layer while the other two have excitatory and inhibitory neurons in both the input and hidden layers. All SNNs have the same numbers of neurons in three layers.

TABLE II
ALGORITHM SETTINGS

Network	Algorithm	Network Structure
Q-learning	Q-learning	Look-up table
S1	A-RM-STDP	E & I in Input layer; E in Hidden layer
S2	M-RM-STDP	E & I in Input layer; E in Hidden layer
S3	A-RM-STDP	E & I in Input and Hidden layers
S4	M-RM-STDP	E & I in Input and Hidden layers

We examine these five approaches by the *success rate*, SR, defined as as the number of successful trials, i.e. ones in which the robot gets to the target without any collision, divided by the total number of trials. We evaluate the success rate for both the learning and testing phase. The average number of steering movements per trial, N_{mv} (the smaller the better), is another indicator of the quality of learning. We consider N_{mv} in the testing phase for the four SNNs, and N_{mv} in the learning phase for Q-learning as it does not perform well in testing.

A. Scenario 1

As shown in Fig. 9 and Fig. 10, the target (red box) is located in the center of a closed environment. The robot navigates from a randomly chosen point within one of the blue circles on the left and right sides. The starting blue circle is also chosen at random. This setup forces the robot to explore the unknown environment widely. 2,000 training and 100 testing trials are used in the learning and testing phase, respectively, for Q-learning. The learning with the SNNs converges much faster so we use only 20 training trials and 30 testing trials. The results are shown in Table III.

Despite of the large number of training trials used, Q-learning performs poorly with an extremely low success rate (SR) in both the learning and testing phase. In addition, Q-learning also leads to a large number of steering movements in the testing phase, indicating a poor quality of trajectories learnt. Note again that in this paper, for Q-learning, N_{mv} is reported based on the training data. The four SNNs demonstrate a much better performance than Q-learning. Among

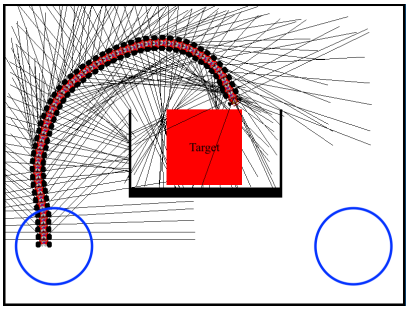

Fig. 9. A simulated trial in the learning phase of S4 for Scenario 1.

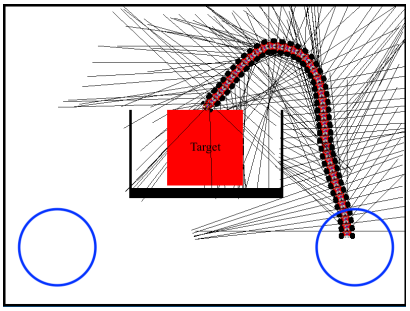

Fig. 10. A simulated trial in the testing phase of S4 for Scenario 1.

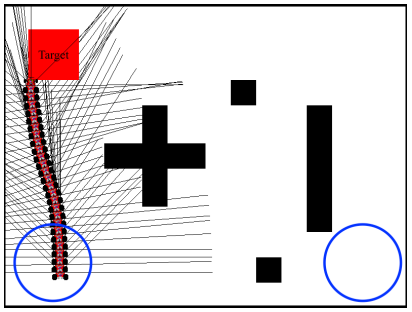

Fig. 11. A simulated trial in the learning phase of S4 for Scenario 2.

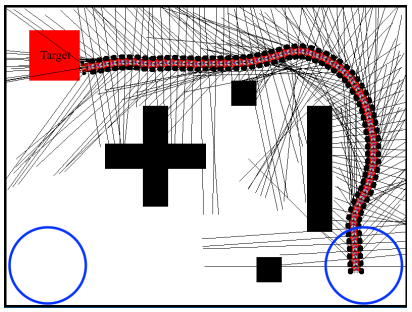

Fig. 12. A simulated trial in the testing phase of S4 for Scenario 2.

them, the proposed multiplicative RM-STDP scheme (M-RM-STDP) outperforms its additive counterpart (A-RM-STDP) rather noticeably in terms of success rate for both learning and testing.

The proposed neural network architecture of Fig. 8, where the hidden layer is composed of an equal number of excitatory and inhibitory neurons, performs fairly well. In particular, the network S4, which combines the proposed M-RM-STDP and network architecture has the highest success rate for both learning and testing, and dramatically outperforms S1, which is the baseline SNN. S4 also produces almost the lowest number of steering movements in testing. Fig. 9 and Fig. 10 show two representative trajectories produced by S4 in the learning and testing phase.

It is interesting to note that S2 somewhat outperforms S3 in terms of success rate while S3 produces a smaller N_{mv}. S2 employs only the proposed M-RM-STDP while S3 only employs the proposed neural network architecture.

B. Scenario 2

The second scenario has obstacles of varying shapes (Fig. 11 and Fig. 12). Again, the robot starts to navigate randomly from two blue circles. The same numbers of trials are used for Q-learning: 2,000 in the learning phase and 100 in the testing phase. For the SNNs, 20 training trials are used, 10 of which start from the left circle and the remaining 10 start from the right circle. 20 trials are used for testing. Table IV summarizes the performances of the five approaches.

TABLE III
RESULT OF SCENARIO 1

Network	SR-Learning	SR-Testing	N_{mv}
Q-learning	4.05%	10%	62
S1:A-RM	45%	70%	50
S2:M-RM	65%	95%	49
S3:A-RM (E+I in hidden)	55%	90%	45
S4:M-RM (E+I in hidden)	70%	100%	46

TABLE IV
RESULT OF SCENARIO 2

Network	SR-Learning	SR-Testing	N_{mv}
Q-learning	4.05%	0%	75
S1: A-RM	75%	45%	67
S2: M-RM	50%	85%	65
S3: A-RM (E + I in hidden)	55%	55%	64
S4: M-RM (E + I in hidden)	40%	90%	62

For this more challenging test case, Q-learning performs even worse, and completely fails the testing. Among the four SNNs, adopting the proposed M-RM-STDP leads to a success rate lower than A-RM-STDP in the learning phase, but a significantly boosted success rate in the testing phase. For example, the testing SR is 55% for S3, which is boosted to 90% by S4. The proposed neural network architecture appears to have a negative impact on the SR in the learning phase, as observed by comparing S3 to S1 and S4 to S2. It, however, noticeably improves the SR during testing. S4, which combines the two proposed techniques, has the highest success rate (90%) and the lowest N_{mv} (62) for testing, showing the best overall performance. We show two simulated trials of S4 in Fig. 11 and Fig. 12.

C. Scenario 3

In this last test case, the robot navigates from randomly sampled points in the blue circle at the bottom-right corner

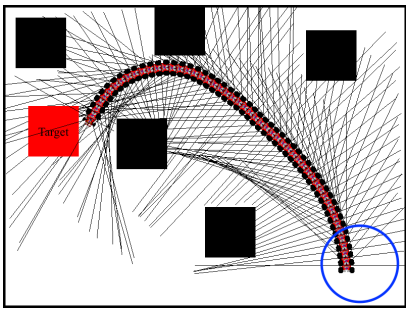

Fig. 13. A simulated trial in the learning phase of S4 for Scenario 3

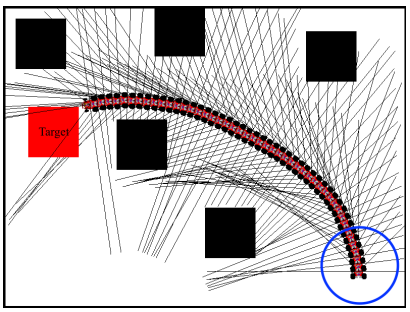

Fig. 14. A simulated trial in the testing phase of S4 for Scenario 3.

of the enclosed environment with more obstacles (Fig. 13 and Fig. 14). 2,000 trials are used for training the Q-learning based robot and 100 trials for testing. The detailed simulation results are described in Table V. Q-learning again performs poorly and is not able to successfully reach the target at all during testing phase. In comparison, the SNNs deliver a much better performance with the network S4 producing the highest success rate (90%) and the lowest N_{mv} (54) during testing. Two simulated trajectories of S4 are shown in Fig. 13 and Fig. 14.

TABLE V
RESULTS OF SCENARIO 3

Network	SR-Learning	SR-Testing	N_{mv}
Q-learning	2.05%	0%	72
S1: A-RM	20%	75%	61
S2: M-RM	40%	90%	57
S3: A-RM (E + I in hidden)	60%	65%	56
S4: M-RM (E + I in hidden)	45%	90%	54

VII. CONCLUSION

This paper proposes two techniques for spiking neural network based reinforcement learning for mobile robot navigation: a new multiplicative RM-STDP (M-RM-STDP) scheme and feed-forward spiking neural network architecture with fine-grained rewards. It has been shown that the proposed techniques significantly outperform Q-learning and a baseline SNN approach. Especially, combining the two proposed techniques leads to a fairly robust solution with significantly improved success rates and quality of navigation trajectories measured by the number of steering movements. In our future work, the two proposed techniques will be demonstrated on a real robot.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. CCF-1639995 and the Semiconductor Research Corporation (SRC) under Task 2692.001.

REFERENCES

[1] J. Gorzelany, "The safest cars and crossovers for 2016," *Forbes*, 2016.
[2] T. Dierks, B. T. Thumati, and S. Jagannathan, "Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence," *Neural Networks*, vol. 22, no. 5, pp. 851–860, 2009.
[3] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
[4] B.-Q. Huang, G.-Y. Cao, and M. Guo, "Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 1. IEEE, 2005, pp. 85–89.
[5] O. Azouaoui, A. Cherifi, R. Bensalem, A. Farah, and K. Achour, "Reinforcement learning-based group navigation approach for multiple autonomous robotic systems," *Advanced Robotics*, vol. 20, no. 5, pp. 519–542, 2006.
[6] J. Qiao, Z. Hou, and X. Ruan, "Application of reinforcement learning based on neural network to dynamic obstacle avoidance," in *Information and Automation, 2008. ICIA 2008. International Conference on*. IEEE, 2008, pp. 784–788.
[7] K. Macek, I. PetroviC, and N. Peric, "A reinforcement learning approach to obstacle avoidance of mobile robots," in *Advanced Motion Control, 2002. 7th International Workshop on*. IEEE, 2002, pp. 462–466.
[8] 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.
[9] L. I. Helgadóttir, J. Haenicke, T. Landgraf, R. Rojas, and M. P. Nawrot, "Conditioned behavior in a robot controlled by a spiking neural network," in *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*. IEEE, 2013, pp. 891–894.
[10] R. V. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neural Computation*, vol. 19, no. 6, pp. 1468–1502, 2007.
[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.
[12] R. Legenstein, D. Pecevski, and W. Maass, "A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback," *PLoS Comput Biol*, vol. 4, no. 10, p. e1000180, 2008.
[13] 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.
[14] P. Arena, L. Fortuna, M. Frasca, and L. Patané, "Learning anticipation via spiking networks: application to navigation control," *IEEE transactions on neural networks*, vol. 20, no. 2, pp. 202–216, 2009.
[15] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.
[16] E. M. Izhikevich *et al.*, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.
[17] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.
[18] J. Sjöström and W. Gerstner, "Spike-timing dependent plasticity," *Spike-timing dependent plasticity*, p. 35, 2010.
[19] M. C. Van Rossum, G. Q. Bi, and G. G. Turrigiano, "Stable hebbian learning from spike timing-dependent plasticity," *The Journal of Neuroscience*, vol. 20, no. 23, pp. 8812–8821, 2000.
[20] Y. Luz and M. Shamir, "Balancing feed-forward excitation and inhibition via hebbian inhibitory synaptic plasticity," *PLoS Comput Biol*, vol. 8, no. 1, p. e1002334, 2012.
[21] R. Evans, "Reinforcement learning in a neurally controlled robot using dopamine modulated stdp," *arXiv preprint arXiv:1502.06096*, 2015.
[22] D. F. Goodman and R. Brette, "The brian simulator," *Frontiers in neuroscience*, vol. 3, p. 26, 2009.