TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

Learning Spiking Neural Controllers for In-silico Navigation Experiments

Mahmoud Akl, Florian Walter and Florian Röhrbein

TUM-I1534

Learning Spiking Neural Controllers for In-silico Navigation Experiments

Mahmoud Akl (mahmoud.akl@tum.de)
Florian Walter (florian.walter@tum.de)
Florian Röhrbein (florian.roehrbein@in.tum.de)
Department of Informatics VI, Technische Universität München
Munich, Germany

Abstract

Artificial neural networks have been employed in many areas of cognitive systems research, ranging from low-level control tasks to high-level cognition. However, there is only few work on the use of spiking neural networks in these fields. Unlike artificial neurons, spiking neuron models are designed to approximate the dynamics of biological neurons. In this work, we developed a virtual environment to explore solving navigation tasks using spiking neural networks. We first used an experimental setup inspired by Floreano and Mattiussi (2001) and compared the results to validate the developed environment. An evolutionary approach is used to set the parameters of a spiking neural network controlling a robot to navigate without collisions. In a second set of experiments, we trained the network via reinforcement learning which was implemented as a reward-based STDP protocol. Our results validate the correctness of the developed virtual environment and demonstrate the usefulness of using such a platform. The virtual environment guarantees the reproducibility of our experiments and can be easily adapted for future research.

Keywords: spiking neural networks; navigation; learning; neurorobotics; simulation; in-silico experiments

Introduction

Work on building cognitive artifacts has always been inspired by biological systems. It is well established that biological brains, whether human or not, process information in a different manner than computers do today. Although Artificial Neural Networks (ANNs) are rapidly progressing, they are still not capable of performing computations on a time scale comparable to a biological brain. ANNs abstract from biological neurons by assuming that neurons communicate real values rather than discrete spikes. From a biological perspective, the real value is interpreted as the firing rate of the neuron. Even with this abstraction, learning algorithms applied to these models have proven to be very successful in certain applications (Mnih et al., 2013; Sutskever, Martens, & Hinton, 2011).

Despite being successful in certain tasks, they remain incompetent compared to human intelligence. In (Nguyen, Yosinski, & Clune, 2014), unrecognizable images to humans were classified as recognizable objects with over 99% confidence. Moreover, in (Szegedy et al., 2013), imperceptible perturbation to test images were shown to possibly alter the network's prediction. The incompetence of ANNs in some domains and the fact that they are not biologically plausible gave rise to Spiking Neuron Models (SNMs), also referred to as the third generation of neural networks (Maass, 1997).

In this work a virtual environment was developed to host experiments for learning spiking neural controllers to solve navigation tasks. The advantages of building an experimental environment for simulations are twofold. Firstly, modifications in the environment are easily worked out, e.g. changing the size and shape of the environment as well as adding obstacles or visual cues. Additionally, experimenters could choose different agents, neural networks, neuronal models, or sensor models to explore with. Secondly, running simulations of an experiment in a virtual environment is much faster than conducting the experiment physically.

In order to investigate the correctness of the developed virtual environment, we first carried out trials of the experimental setup described in (Floreano & Mattiussi, 2001). This experiment shows superiority of evolving Spiking Neural Networks (SNNs) over ANNs to solve a very simple autonomous navigation task. Furthermore, we tested the effects of modifying some parameters of the original experiment, e.g. neuronal model and initial positions, on the results. We also carried out the same experiment in the virtual environment using a reinforcement learning model instead of the evolutionary approach originally used.

To simulate spiking neurons, the Neural Simulation Tool (NEST) was used (Gewaltig & Diesmann, 2007). Even though it contains several neuronal and synaptic models, it does not include a reinforcement learning model for spiking neurons. The foundations for the model found in (Izhikevich, 2007) have therefore been implemented and the effectiveness of the implemented model has also been tested in first experiments.

The next section gives a brief background of spiking neural networks with emphasis on the neuronal models used in the experiments conducted here.

Background

The characteristics of biological neurons form the basis for artificial and spiking neural models. The state of a neuron is described by the potential difference across its membrane. At rest, a neuron is negatively polarized and its membrane potential v_{rest} is approximately -70mV. A spike is emitted only when the neuron's membrane potential crosses a threshold θ. Typically, spikes last about 1-2 ms and have an amplitude of $\sim$ 100mV. A spike's effect on the postsynaptic neuron's membrane potential, also called the Postsynaptic Potential (PSP), depends on the properties of the synapse and on the time the spike took to reach the postsynaptic neuron. If the PSP causes the postsynaptic neuron to depolarize, then it is referred to as Excitatory Postsynaptic Potential (EPSP). Otherwise, if it causes the postsynaptic neuron to hyperpolarize, then it is referred to as Inhibitory Postsynaptic Potential (IPSP). Figure 1 illustrates the difference between EPSP and

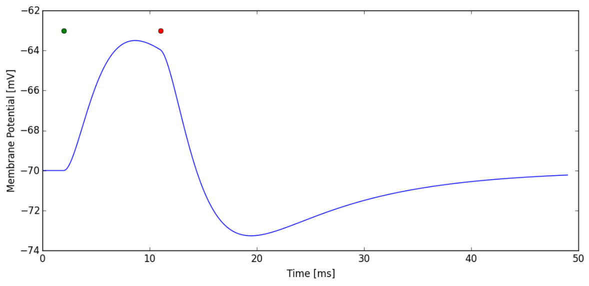


Figure 1: The first dot indicates the arrival of a spike from an excitatory synapse. The EPSP depolarizes the neuron by raising its membrane potential from its resting value to ~ -64mV. The second dot indicates the arrival of a spike from an inhibitory synapse. The IPSP hyperpolarizes the neuron by dropping its membrane potential below its resting value.

IPSP.

After a neuron has emitted a spike, it enters absolute refractoriness for a few milliseconds, where it is impossible to emit another spike right away. After that, the membrane potential recovers gradually to its resting value. This recovery period is referred to as the relative refractory period, where it possible but hard for a neuron to fire.

Several SNMs have been developed that hold to the nature of biological spiking neurons. They are, however, considered to be simplified models because they do not imitate biological neurons in full detail, like the Hodgkin-Huxley model for example. Due to the facts that spikes are the only method of communication between biological neurons, and that all spikes have the same characteristics (amplitude and duration), they cannot possibly convey important information. Therefore the exact timing of spikes must play the important role of encoding data. In the experiments conducted here, two neuronal models were used.

The first model examined was the Leaky Integrate-and-Fire (LIF) model with alpha-shaped synaptic currents. The LIF model has a refractory period parameter t_{ref} which defines, in milliseconds, the duration of the refractory period. During this period no spikes are emitted by the neuron and the membrane potential is clamped to a predefined value V_{reset}. The differential equation defining the dynamics of the membrane potential in the LIF model behavior is

$$\frac{d}{dt}V_m = \frac{-(V_m - E_L)}{\tau_m} + \frac{I_{syn}(t)}{C_m} \quad (1)$$

where E_L is the resting membrane potential, C_m is the capacity of the membrane, τ_m is the membrane time constant and $I_{syn}(t)$ denotes the sum of the input synaptic currents. In the absence of input synaptic currents ($I_{syn} = 0$), the membrane potential gradually returns to the resting potential E_L.

The second model used was the Multi-timescale Adaptive Threshold (MAT) model developed by (Kobayashi, Tsubo, & Shinomoto, 2009). In this model, the membrane potential never resets. Instead, after the neuron emits a spike, the threshold jumps to a very high value and then gradually returns to its resting value according to

$$\theta(t) = \sum_j \alpha_1 e^{t_j - t/\tau_1} + \omega \quad (2)$$

where ω is the threshold's resting value, α_1 is the weight, t_j denotes spike times and τ_1 is the time constant with which the threshold decays back to its resting value.

In the next section, the characteristics of the virtual environment and the setup of the experiments will be detailed. The two following sections will discuss the results of evolving and learning the spiking controllers in this experimental setup.

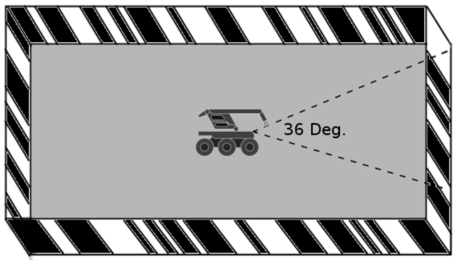


Figure 2: The virtual arena where experiments took place viewed from above. The robot is placed at the center of the arena and the dashed lines represent the view borders according to the 36° view angle.

Experimental Setup

The virtual environment in which the experiments took place is a rectangular arena with black and white vertical stripes painted on the walls. The widths of the stripes were drawn randomly from the range [0.5 - 5] cm. The dimensions of the arena were chosen to be 687x371 mm^2.

A two-wheeled robot was placed at random initial poses in the arena. The robot was equipped with a camera module consisting of 64 photoreceptors and permitting a visual angle of 36°. The experimental setup is seen in Figure 2.

A SNN of fixed size, consisting of 10 neurons and 18 receptors, was interfaced to the robot. 16 of the receptors transmitted vision signals, while the remaining two transmitted the error between the desired and the actual speeds of the wheels. Each wheel was assigned two neurons, one setting the forward speed while the other sets the backward speed. The overall wheel speed was then determined by the algebraic sum of both speeds, which was then mapped to a maximum speed of 80 mm/s. The four neurons controlling the wheels' speeds are referred to as *motor neurons*. An illustration of the neural network is seen in Figure 3.

An experiment's duration was 40 seconds divided into 100 ms chunks. At the beginning of each 100 ms interval the photoreceptors' values are preprocessed and used to set the firing probabilities of the receptors. The neural network, how-

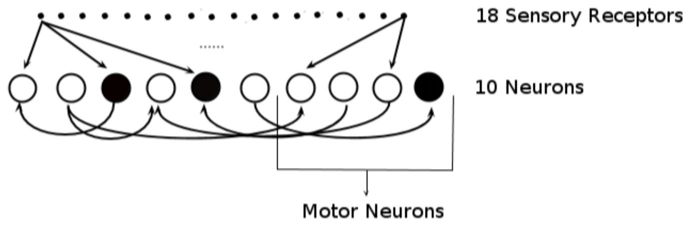


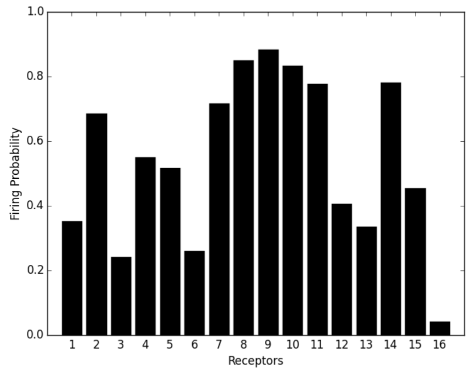
Figure 3: Illustration of the spiking neural network with arbitrary connections. Black circles indicate excitatory neurons and white circles indicate inhibitory neurons.

ever, gets only 16 visual signals as input. To achieve this, 16 equally spaced pixel values are read from the receptors. Uniform random noise is then added to the pixel values in an attempt to model the irregularities of the physical world. The noisy values are then convolved with a Laplace filter spanning three adjacent photoreceptors in order to detect contrast. Finally, the absolute resulting values are scaled to the range [0, 1] and used to set the probabilities of the corresponding 16 neural receptors to emit a spike at that time step. Based on this process, high stripe frequency results in high firing probabilities. In Figure 4, the 16 visual receptors' firing probabilities are shown for high and low stripe frequencies. Visible stripe frequencies depend on the robot's pose in the arena.

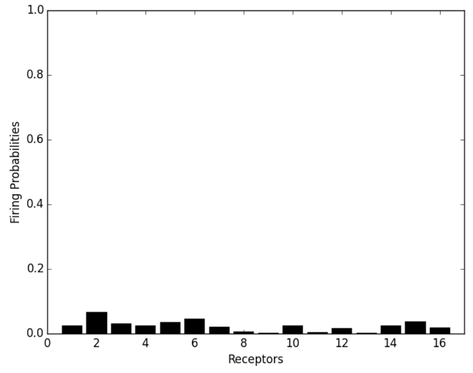
After setting the probabilities for the neural receptors to emit a spike at the corresponding time step, the neurons communicate via spikes during the 100 ms interval and at the end of the interval the wheels' speeds are set by reading out the firing rates of the motor neurons. These speeds remain constant during the whole interval. The robot then moves according to the assigned speed to each wheel. A kinematic model based on differential drive is used to translate the wheel speeds (v_l, v_r) into linear and angular speeds and eventually into the robots new pose (x, y, θ). If the new pose happens to collide with one of the walls, then the robot remains stuck in this pose for the remaining time of the simulation.In Figure 5 the trajectory and speed profile of a colliding robot are shown to illustrate this behavior.

The speeds of the wheels, which are read out from the motor neurons, represent the desired values. To simulate an error between the desired and the actual speeds, each wheel speed read out from the corresponding motor neurons was multiplied by a uniform random variable in the range [0.7, 1]. The error signals were also fed as input to the neural network, setting the probabilities of the two remaining receptors to fire.

The MAT and LIF neuronal models were parameterized with the values found in Table 1 and Table 2 respectively. Noise was added to the models' refractory functions at each timestep. In the LIF model, the parameter t_{ref} was set to a uniform random variable in the range [0 - 1], while the parameter V_{reset} was also set to a uniform random variable in the range [-0.1 - 0]. In the MAT model, however, the parameter α_1 was set to a uniform random variable in the range[0 - 1] at each time



(a) Distant view



(b) Close view

Figure 4: Probabilities of the 16 visual receptors of emitting a spike when multiple stripes are visible (distant view) and when only one stripe is visible (close view).

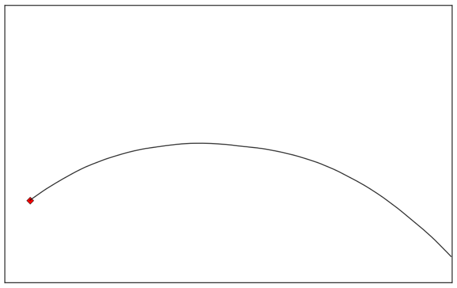
step.

Results Evolutionary Learning

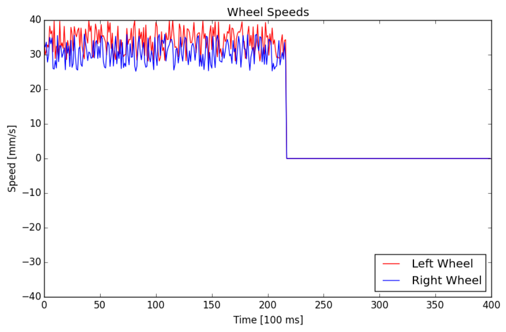
In the evolutionary experiments, the connections of the network were encoded in a binary genetic string consisting of 290 bits. The genetic string was divided into 10 blocks, one for each neuron. The first sign of each block indicates whether a neuron is excitatory or inhibitory. The remaining 28 bits of the block indicate the presence or absence of connections from the 10 neurons and the 18 receptors to the corresponding neuron of that block. In the experiments conducted here, sensory receptors were always excitatory. An illustration of the connections encoded in an arbitrary genetic

Table 1: Values assigned to the MAT model parameters in NEST during the simulations.

E_L	τ_m	τ_{syn}	V_m	C_m	τ_1	α_1	ω
0.0	4.0	3.0	0.0	10.0	4.0	1.0	0.1



(a) Trajectory during simulation. The red dot indicates the initial position and the black line is the path of the robot during simulation.



(b) Speeds of the left and right wheel at each time-step during the simulation. The speeds decrease to zero after collision.

Figure 5: Illustration of a collision scenario. The robot collides and remains stuck for the remaining simulation time.

Table 2: Values assigned to the LIF model parameters in NEST during the simulations.

E_L	τ_m	τ_{syn}	V_m	C_m	t_{ref}	V_{th}	V_{reset}
0.0	4.0	1.0	0.0	10.0	1.0	0.1	-0.1

string is shown in Figure 7.

Three populations were generated randomly, each containing 60 individuals, and were evolved for 30 generations using one-point crossover, bit-mutation and elitism. The fitness function used was an averaged sum of both wheel speeds at all time-steps. A comparison of the fitness values achieved by both neuronal models tested is seen in Figure 6. Both models achieve similar average fitness results, but the MAT model achieves better best fitness results. A comparison between the resulting paths of the best performing individual after 30 generations of both models is seen in Figure 9.

The evolved fitness values were found similar to those found in (Floreano & Mattiussi, 2001), even though a different neuronal model was used. Other behaviors were also found to be similar. For example, it was observed, by checking the firing rates of the neurons (see Table 3), that the neural network keeps one wheel rotating forwards with constant speed, while controlling the turning angle of the robot by setting the speed of the other wheel. This behavior was also described in (Floreano & Mattiussi, 2001). Moreover, because populations were randomly generated binary strings, they showed initial average connectivity of 50%. This value, however did not change much as the population evolved, which was also mentioned in (Floreano & Mattiussi, 2001).

Further evolutionary experiments were run in different setups to see how changing the setup affects the results. For example, two sets of experiments were run to test whether always starting the evolutionary experiment from a fixed pose is different from starting it from a random initial pose at each run. While the evolved populations' fitness value indicate that a fixed initialization is better (see Figure 8), the best evolved individual was not able to navigate in the environment when starting from a different pose.

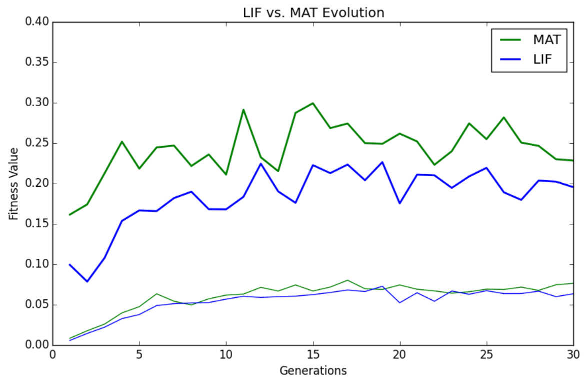


Figure 6: Comparison between fitness values achieved by both neuronal models. Thin lines represent average fitness per generation. Thick lines represent best fitness per generation.

Table 3: Average firing rates of the 10 neurons of the best individual after 30 generations using the MAT model in *arena1*. Neuron 10 is responsible for setting the backward speed of the right wheel.

Neuron	1	2	3	4	5	6	7	8	9	10
Spikes/Sec.	490	382	294	349	5	177	489	359	498	0

Results Reinforcement Learning

The reinforcement learning model implemented here is described in detail in (Izhikevich, 2007). Weights are updated whenever rewards are issued according to the equation

$$\frac{d}{dt}w_{ji}(t) = c_{ji}(t)d(t) \tag{3}$$

where $c_{ji}(t)$ is the eligibility trace between neurons i and j and $d(t)$ is the reward signal. The value of the eligibility trace depends on the firing of the pre- and postsynaptic neurons and

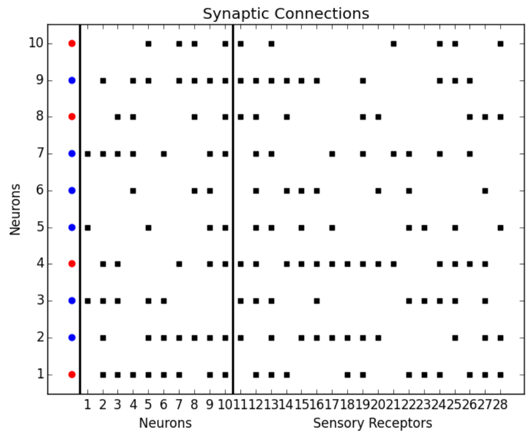


Figure 7: Illustration of connections encoded in a binary genetic string. The black squares indicate the presence of a synaptic connection. Red dots indicate inhibitory outgoing connections from the corresponding neuron while blue dots indicate excitatory ones.

is updated based on the Spike Timing Dependent Plasticity (STDP) rule. The dynamics of the eligibility trace are defined by the equation

$$\dot{c} = -c/\tau_c + \mathrm{STDP}(\tau)\delta(t - t_{pre/post}) \tag{4}$$

Here, δ is the Dirac delta function that step-increases the variable c, where $t_{pre/post}$ are the firing times of pre- and postsynaptic neurons respectively. The magnitude of change in the variable c is determined by the STDP rule, which in turn is influenced by the inter-spike interval $\tau = t_{post} - t_{pre}$. The eligibility trace decays to zero exponentially with the time constant τ_c. Positive inter-spike intervals indicate causal firing between the pre- and the postsynaptic neurons and hence amplify the variable c. Negative inter-spike intervals, however, indicate acausal firing, i.e. the postsynaptic neuron emits a spike before the presynaptic neuron does, and hence decrease the value of c.

In the reinforcement learning experiments, the network was fully connected, i.e. there were 280 synaptic connections. Initial weights were randomly chosen from a uniform distribution in the range [0 - 1.5] for the connections between receptors and neurons, and in the range [-1.5 - 1.5] for interneuron connections. Maximum values for weights were chosen to be -3.0 and 3.0 for inhibitory and excitatory synapses respectively.

The experiment consisted of 10 runs, each lasting for 40 seconds and divided into 400 chunks just as in the evolutionary experiments. After 100 ms of simulation, when the speeds of the wheels are set for the next interval, the eligibility traces for all 280 connections were also calculated based on the spike trains of the previous 100 ms.

Rewards, however, were issued every second of simulation. The reward signal is analogous to the fitness function described in the evolutionary experiments. It is the averaged sum of wheel speeds over the previous second of simulation. If the wheels were not moving or rotating backwards, then a negative reward was issued to penalize the behavior. Moreover, if the wheels were moving forwards but the robot ended up colliding with one of the walls, then a negative reward was also issued.

After one run was completed, the robot was returned to its initial position to start another run without modifying the weights adapted so far. In Figure 10 the trajectories of the first and tenth run are shown. After 10 runs the robot's path is much more smooth and achieves, in an evolutionary sense, a higher fitness value.

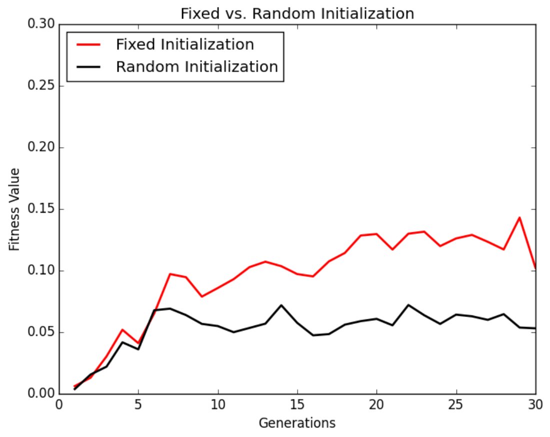


Figure 8: Comparison between fitness values achieved when starting from fixed and random poses.

Discussion and Outlook

The aim of this work was to develop a virtual environment that should serve as a platform for testing learning techniques with various SNMs in navigation experiments. Validation of the virtual environment was done by conducting the same experiment found in (Floreano & Mattiussi, 2001) and observing huge similarities in the results. There were, however, small differences in the results, which are attributed to the subtle differences between both setups. For example, the exact size of the arena used in the original experiment was not mentioned, and different neuronal models were used. Furthermore, different experimental setups were explored to study their effects on the results. Based on these explorations it was concluded that starting the evolutionary experiments from fixed poses achieves higher fitness values but does not generalize. Such explorations and findings are much easier to be carried out in a virtual environment. For example, a simulation of a 40 second experiment takes only 4 seconds to complete.

Moreover, the basis for reinforcement learning has been laid in this work by the implementation of the eligibility trace between neurons. The implementation of the model has also

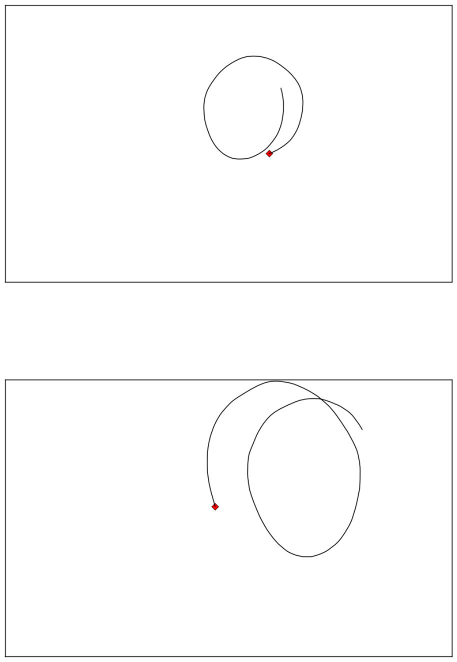


Figure 9: Comparison between paths of the best performing individuals after 30 generations using LIF (top) and MAT (bottom) neuronal models.

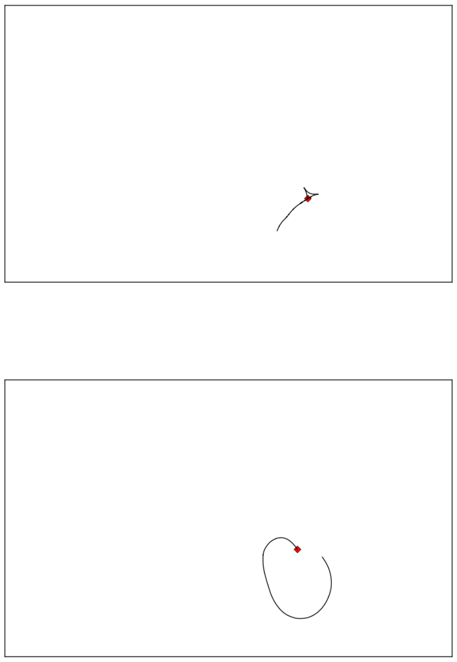


Figure 10: Comparison between trajectories of the first (top) and tenth (bottom) run when applying reinforcement learning.

been tested in a first experiment that proved the effectiveness of the model. Even though the experiment showed that the robot started adapting to the desired behavior after 10 runs, more thoroughly designed experiments are required to explore the reinforcement learning technique in the task at hand. For example, the frequencies with which rewards are issued were chosen arbitrarily in this experiment.

To conclude, the developed environment provides a suitable basis for the development of a sophisticated platform that allows for testing behavior adaptation techniques using SNNs, and the implemented reinforcement learning model provides a suitable basis for a comparison between the effectiveness of evolutionary algorithms and reinforcement learning for solving an autonomous navigation task.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 604102 (HBP).

References

Floreano, D., & Mattiussi, C. (2001). Evolution of spiking controllers for autonomous vision-based robots. In T. Gomi (Ed.), *Evolutionary robotics. from intelligent robotics to artificial life* (Vol. 2217, p. 38-61). Springer Berlin Heidelberg.

Gewaltig, M.-O., & Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, *2*(4), 1430.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral cortex*, *17*(10), 2443–2452.

Kobayashi, R., Tsubo, Y., & Shinomoto, S. (2009). Made-to-order spiking neuron model equipped with a multi-timescale adaptive threshold. *Frontiers in computational neuroscience*, *3*.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural networks*, *10*(9), 1659–1671.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nguyen, A., Yosinski, J., & Clune, J. (2014). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *arXiv preprint arXiv:1412.1897*.

Sutskever, I., Martens, J., & Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 1017–1024).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.