Training a Spiking Neural Network using R-STDP to perform Autonomous Target Tracking on a Snake Car Robot

René Romen

September 24, 2018

Chair of Robotics, Artificial Intelligence and Real-time Systems
Department of Informatics
Technical University of Munich

Contents

SNN and the Planar snake car model

- Spiking Neuronal Networks are promising for robotics
- **BUT** they can’t be trained using gradient descend methods
- Snake-like robots are very mobile
- Simple planar snake robot with slithering gait model and wheels
- Highlevel controll using SNN

Task: Target Tracking

- Target Tracking

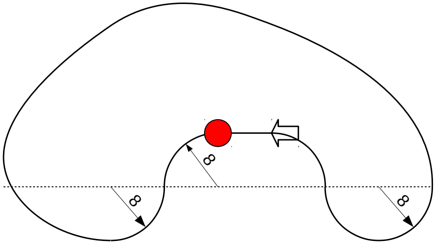

Figure 1: Target tracking SNN evaluation environment.

Task: Target Tracking

- Target Tracking
- Prevent collisions with walls

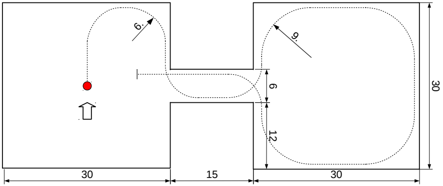

Figure 2: Evaluation environment

Target Following SNN

- Infrared image input 16×4 pixel resolution

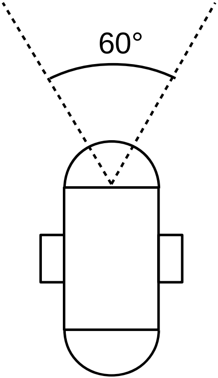

Figure 3: Infrared vision sensor

Chair of Robotics, Artificial Intelligence and Real-time Systems
Department of Informatics
Technical University of Munich

Target Following SNN

- Infrared image input 16×4 pixel resolution
- Image preprocessing

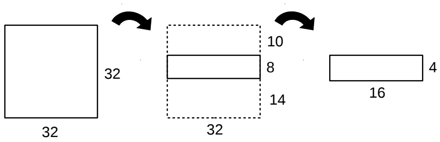

Figure 4: Image preprocessing in 3 steps

Chair of Robotics, Artificial Intelligence and Real-time Systems
Department of Informatics
Technical University of Munich

Target Following SNN

- Infrared image input 16 × 4 pixel resolution
- Image preprocessing
- 64 Poisson input neurons
- Feed forward architecture
- Left and Right LIF output Neurons

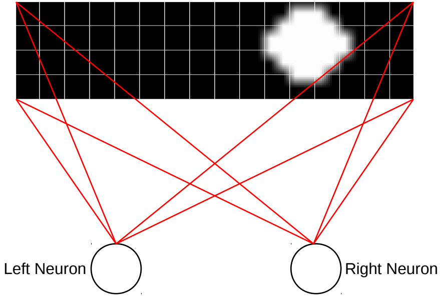

Figure 5: Target following SNN architecture

Target Following SNN cont.

- Output interpreted as angle

$$decode\left(n_{spikes}\right) = \frac{n_{spikes}}{n_{max}}$$

$$\alpha = \alpha_{max}\left(n_l - n_r\right)$$

$$\alpha_t = c\alpha + \left(1 - c\right)\alpha_{t-1}$$

Chair of Robotics, Artificial Intelligence and Real-time Systems
Department of Informatics
Technical University of Munich

Target Following SNN cont.

- Output interpreted as angle
- Reward depends on Angle between head module and target

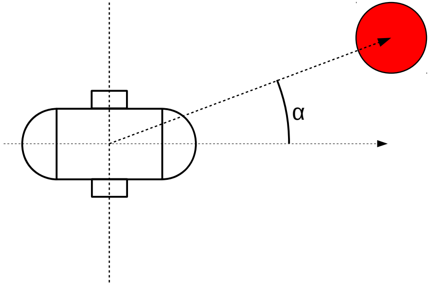

Figure 6: Angle between robot head module and target.

Target Following SNN cont.

- Output interpreted as angle
- Reward depends on Angle between head module and target
- Left and right neuron get the opposite rewards of each other

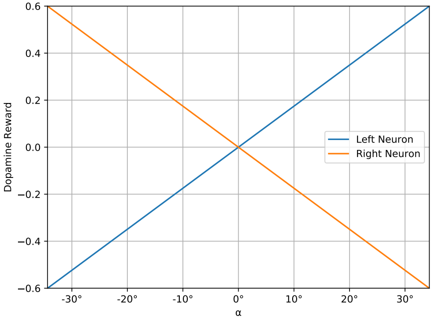

Figure 7: Target following reward function

Obstacle Avoidance SNN

- Four proximity sensors

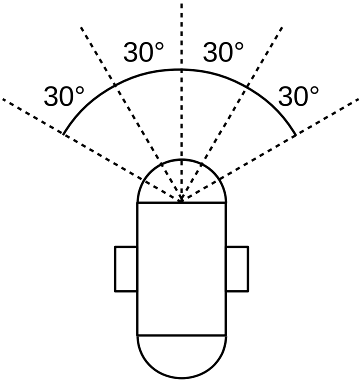

Figure 8: Proximity sensors

Obstacle Avoidance SNN

- Four proximity sensors
- Proximity data preprocessing

- Data in range $[0; 3]$
- Mapped to range $[0 : 1]$
- 0: No obstacle or at maximum distance
- 1: Close obstacle

Obstacle Avoidance SNN

- Four proximity sensors
- Proximity data preprocessing
- 4 Poisson input neurons
- Feed forward architecture
- Left and Right LIF output Neurons

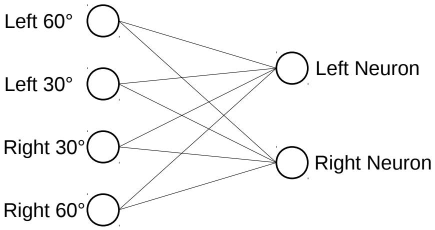

Figure 9: Obstacle avoidance SNN architecture

Obstacle Avoidance SNN cont.

- Output interpreted as angle

$$decode\left(n_{spikes}\right) = \frac{n_{spikes}}{n_{max}}$$

$$\alpha = \alpha_{max}\left(n_l - n_r\right)$$

$$\alpha_t = c\alpha + \left(1 - c\right)\alpha_{t-1}$$

Obstacle Avoidance SNN cont.

- Output interpreted as angle

$$decode\left(n_{spikes}\right) = \frac{n_{spikes}}{n_{max}}$$

$$\alpha = \alpha_{max}\left(n_l - n_r\right)$$

$$\alpha_t = c\alpha + (1 - c)\,\alpha_{t-1}$$

- Event based rewards on Episode failure
- Left and right neuron get the opposite rewards of each other
- 4 Reward cases, collision and target lost, obstacle left or right side

Training

- Training environments

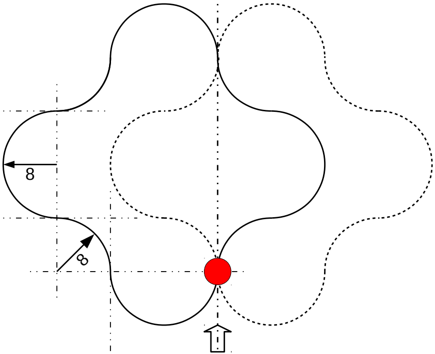

Figure 10: Target tracking SNN training path.

Training

- Training environments

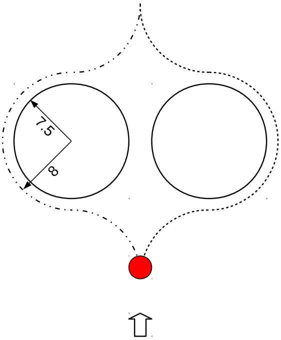

Figure 11: Obstacle avoidance SNN training path.

Chair of Robotics, Artificial Intelligence and Real-time Systems
Department of Informatics
Technical University of Munich

Evaluation

- Average error e = 7,39°

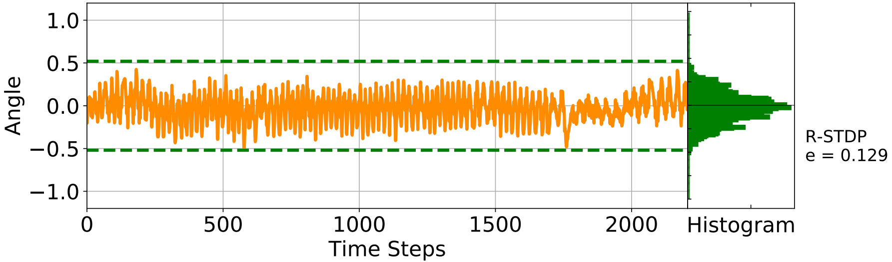

Figure 12: Performance on Target Following Task

Evaluation

- Average error $e = 7{,}39°$
- Average error $e = 8{,}71°$

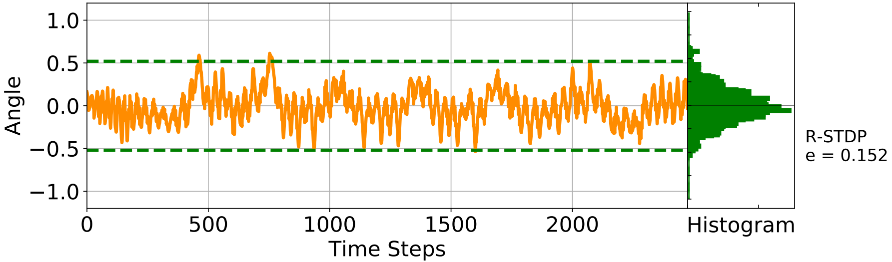

Figure 13: Performance on Target Tracking and Obstacle Avoidance Task