ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

Short-term trajectory planning using reinforcement learning within a neuromorphic control architecture

Florian Mirus[1,2], Benjamin Zorn[1,2] and Jörg Conradt[3]

1- BMW Group - Research, New Technologies, Innovations, Garching, Germany
2- Technical University of Munich - Department of Electrical and Computer Engineering, Munich, Germany
3- KTH Royal Institute of Technology - Department of Computational Science and Technology, Stockholm, Sweden

Abstract. In this paper, we present a first step towards neuromorphic vehicle control. We propose a modular and hierarchical system architecture entirely implemented in a spiking neuron substrate, which allows for the adjustment of individual components through either supervised or reinforcement learning as well as future deployment on dedicated neuromorphic hardware. In a sample instantiation, we investigate automated training of a neuromorphic trajectory selection module using reinforcement learning to demonstrate the general feasibility of our approach. We evaluate our system using the open-source race car simulator TORCS.

1 Introduction

Increasing energy requirements of driver assistance systems and automated driving functions, oftentimes built upon modern machine learning techniques, demand for new approaches to satisfy these power demands with limited on-board resources. SNNs (Spiking Neural Networks) offer an interesting option to tackle this issue when deployed on dedicated, energy-efficient neuromorphic hardware. The authors of [1] show on the example of image classification using CNNs (Convolutional Neural Networks), that such networks can effectively be converted to SNNs and that the converted networks can be significantly more energy-efficient when run on specialized neuromorphic hardware with minimal drops in accuracy compared to the original network.
In this paper, we propose a neuromorphic system for vehicle control. Our system is implemented entirely in a spiking neuron substrate using the Nengo simulator [2] and is designed to be both distributed and hierarchical. In a sample instantiation, we train a trajectory selection module using reinforcement learning to investigate the feasibility of a learnable, neuromorphic control system in an automotive context. This approach has been selected in order to be decoupled from the quality of training data in this first investigation. We evaluate our approach in TORCS (The Open Racing Car Simulator), which allows us to generate training data in a safe and controlled simulation environment. Furthermore, TORCS offers an advanced vehicle physics simulation as well as a variety of sensors and actors for interaction [3].

Related work: There exists a large variety of different approaches to autonomously control a car in TORCS which are either manually engineered or automatically learned from data. Most hand-crafted methods use pre-defined control behaviors and vary in the methods to switch between those controllers, e.g. heuristics [4] or finite-state machines [5]. Oftentimes, these methods are combined with evolutionary or genetic algorithms

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

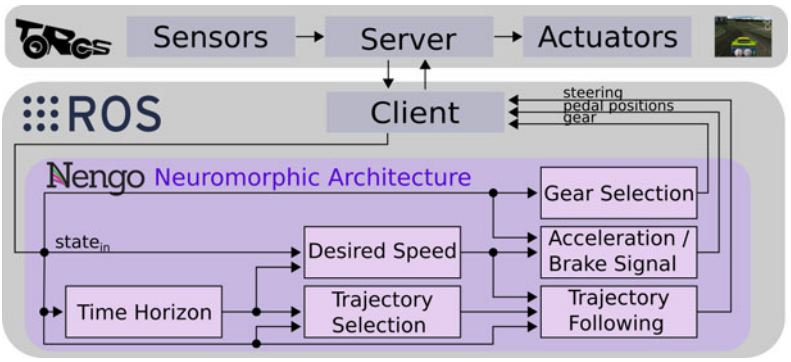


Fig. 1: Proposed distributed neuromorphic architecture utilizing individual modules for separate control signal calculation.

[6] for parameter optimization.
On the other hand, data-driven approaches for vehicle controllers in TORCS using either supervised learning or reinforcement learning are also frequent. Imitation learning methods try to directly learn control signals from another, either human or artificial, driver using a feed-forward neural network [7] or an evolutionary network of spiking Izhikevich neurons [8]. Using reinforcement learning like the asynchronous advantage actor-critic approach [9] to derive end-to-end control policies from (visual) sensory input avoids the need of other drivers to create training examples.
An example of temporal difference learning in spiking neurons is presented in [10], with the introduction of *dual learning*, which solves the issue of spiking networks when a temporal component is included. Neuromorphic control systems for robot motion and manipulation using SNNs have been examined (e.g. in [11]).

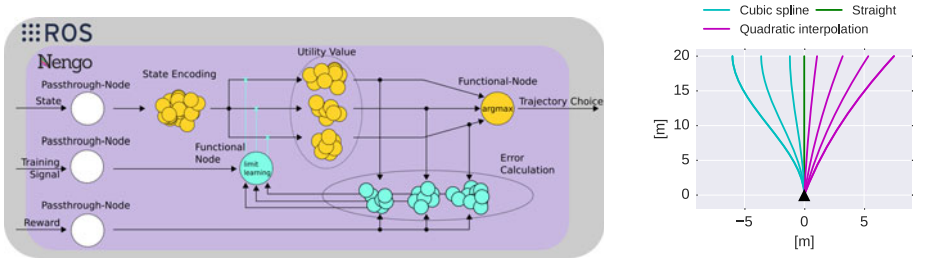
2 System architecture

The architecture of our proposed system is visualized in Fig. 1. Interactions with the TORCS environment (for detailed information on the sensor and actuator setup see [12]) are channeled over several ROS (Robot Operating System) nodes used for the gateway communication. For evaluation and training purposes, a controller using the global position and orientation of the vehicle is implemented in ROS to determine control signals to follow a given trajectory or the roadways centerline at a fixed speed[1].

Distributed neuromorphic architecture: The proposed architecture for a learnable and energy-efficient vehicle control system is a holistic neuromorphic approach for determining steering, gas and brake pedal as well as gear signals. It is a distributed system in that these signals are calculated separately in different modules. In addition, it is a hierarchical system as several intermediate values have to be calculated and different modules and subsystems are dependent upon each other. The modules' vertical alignments within Fig. 1 indicate the distinction of three core subsystems, each responsible for one of the control signals (*gear selection*, *breaking/acceleration* and *steering*). We

[1]The TORCS-to-ROS interface can be found at `https://github.com/fmirus/torcs_ros`. The Nengo and ROS implementation of the trajectory selection module and the framework that is built upon it can be found at `https://github.com/fmirus/torcs_neural_trajectory_ctrl`

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.



(a) Simplified visualization of the SNN for *trajectory selection*. The yellow (*A*) resp. cyan networks (*B*) implement action selection resp. weight optimization for three exemplary trajectory choices

(b) Visualization of a set of exemplary trajectories.

Fig. 2: SNN for *trajectory selection* and exemplary set of trajectories

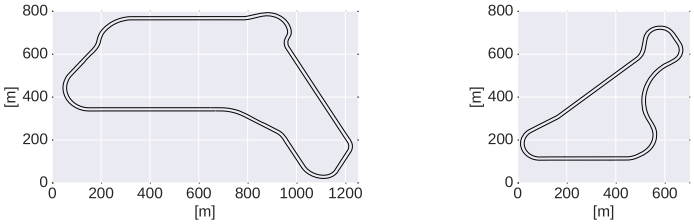
envision each of these submodules to be learnable either by supervised or reinforcement learning. Some of the depicted modules are heavily dependent on each other and will have to be trained in parallel rather than independently.
In this paper, we focus on the *trajectory selection* module to be the only learnable module for simplicity. Therefore, gear selection and acceleration are determined from the engine's rpm and a desired velocity set to a fixed value for now. The *trajectory following* module determines the control values for steering from the chosen trajectory. We envision the *time horizon* module to estimate the time window for the next trajectory to be followed. Assuming a perfect control algorithm and the desired speed to be the actual speed, this equates to determining the trajectory's end point's longitudinal position. Accordingly, the *trajectory selection* module is an action selection module for the agent to choose a trajectory type as well as the lateral component of the trajectory's end point from a set of predefined options.

Trajectory selection module: In this paper, we are focusing on the *trajectory selection* isolated from the other modules. Therefore, we use a classical controller to steer the vehicle along the chosen trajectories at a fixed velocity of $34\,\mathrm{km\,h^{-1}}$ with the trajectories' end points' longitudinal component set to 20 m which emulates a simplified version of *trajectory following*.
We implemented a set of 15 trajectories to describe varying degrees of different behaviors (see Fig. 2b). We use straight lines to emulate lane following, cubic splines (with their derivative set to 0 at the extremities) to model lane change maneuvers and quadratic interpolations between a start and end point to perform a change in orientation. The latter trajectories are intended for driving curves or to correct slight misalignments between the vehicle and the road.

3 Neuromorphic reinforcement learning

Learning network: The core of this work is the learning Spiking Neural Network for *trajectory selection*, which is visualized in Fig. 2a and consists of two subnetworks. It was built using the Nengo neural simulator [13], which implements the principles of the NEF (Neural Engineering Framework) [14]. The NEF provides mathematical tools to construct biologically plausible, large-scale neural models. Time-varying real-valued

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.



(a) Training (b) Validation

Fig. 3: Visualization of the chosen training and validation tracks.

vectors can be encoded in the activity of neural populations, i.e. for each neuron within a group there is a function $a(x)$ that maps a variable x to a neural activity. Expanding on this idea, we can also approximate functions across connections between populations of neurons.

The first subnetwork A (yellow components in Fig. 2a) encodes the current state in a neural population. This population feeds its output to an array of downstream populations, each representing the utility value Q associated to one of the possible actions. Unless an exploration step is enforced, the highest filtered utility value determines the next action to be performed. As we want to adapt the mapping between input (state) and utility values by learning the respective connection weights, their decoders are initialized to a fixed value. The second subnetwork B (cyan components in Fig. 2a) encodes the reward received from the environment (cf. eq. 1). From this reward, the offset to the current utility values is calculated, which in turn is used to adapt the weights of subnetwork A's learning connection.

Associative value learning: The associative learning process is implemented using the PES (Prescribed Error Sensitivity) [13] learning rule, which modifies connection weights based on a (multi-dimensional) error signal $\mathbf{e}$. We calculate the error from the reward function

$$r(t) = |p(t)|\,\beta_1 \cdot |\theta(t)|\,\beta_2 + |p(t) - p(t-1)|\,\beta_3 + |\theta(t) - \theta(t-1)|\,\beta_4, \qquad (1)$$

where p describes the vehicle's lateral position along the road at successive time steps t and $t-1$ (a value of 1 being centered and 0 being at the track boundary), θ describes the vehicle's orientation w.r.t. the road's centerline (values of 1 resp. 0 indicate parallel resp. perpendicular alignment) and β_k being scalar weights. Thus, the reward function is designed to blend the two objectives of driving aligned with and centered on the road. With each ensemble representing a utility value $Q(s,a_j)$ of the state s when taking action a_j, we define the error for dimension j as

$$e_j = \begin{cases} r(t) - Q(s(t),a_j) & \text{if } a_j \text{ is selected} \\ 0 & \text{else.} \end{cases} \qquad (2)$$

For this paper, we have chosen to limit the agent's knowledge to immediate rewards. Therefore, we neglect the correlation between the *trajectory selection* and *time horizon* modules, as it will introduce a temporal component to the discretization. Hence, the training procedure needs to be adjusted once both modules are trained jointly. Naturally,

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

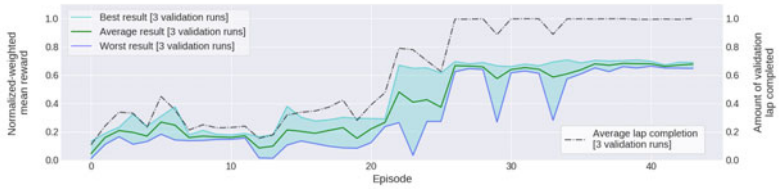


Fig. 4: Results (mean reward and lap completion) for three validation runs per episode

such a limited approach will impair the quality of our results. Our goal in this paper, however, is to show general feasibility of our approach which is why we postpone the coupling with other distributed modules to future work.

Training procedure: In order to evaluate the derived policy's performance, we chose one training and one validation track (Fig. 3) that show similar characteristics in road width and occurring curvatures. They, therefore, show comparable features while being distinct enough to identify if a straight mapping to the training track were to happen. Whenever the vehicle leaves the track during training, the simulation is reset and a reward of 0 is awarded as the track sensors become unreliable in such a scenario. After every five training runs, three runs until the vehicle goes off-track are driven on the validation course without updating any weights. This process constitutes one training/-validation episode.
We use the percentage of completion on the validation track multiplied by the collected average reward in that validation episode as a measure of reliability performance. This measure is additionally normalized by the maximally achievable reward for the utilized parameter set. As the reset leads to a large bias in the experience of states close to the starting line during training, several cyclic points of interests have been defined upfront as substitute starting points before which a non-neuromorphic controller handles the vehicle. This procedure ensures a large exploration of different possible states and accelerates learning. During training, a lineraly decaying epsilon exploration is utilized.

4 Results

Fig. 4 visualizes the performance of the derived control policy on the validation track after each episode. The average performance tends to increase, indicating an overall improvement of the policy given the state values approximated by the SNN. The neuromorphic controller is able to reliably complete the validation lap after roughly 35 episodes without having performed any training on it. This indicates that the SNN is able to learn a meaningful policy and to generalize beyond the training track given the sensors' uncertainties.

5 Discussion

Our investigation in this paper shows promise to be a first step in the direction of a neuromorphic vehicle control. Although we chose a very limited approach in associative reinforcement learning, our system is able to fulfill the initial requirement to complete laps on the unknown validation track. However, tests on other validation tracks revealed issues with this approach. Successive curves are a problematic situation as the vehicle manages to pass the first curve but oftentimes maneuvers itself into a position where it is impossible to complete the second curve with the available trajectories. This problem

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

can not be identified timely, and therefore solved with the current approach, as there is no temporal component in associative reinforcement learning.

Future work: We envision to tackle the temporal issue using Q-learning with *dual learning*. Another direction for future work is to couple the *trajectory selection* with the *time horizon* module to enable planning over time while remaining within a discretized domain and delaying the continuous control to the *trajectory following* module. These modules could be trained separately or jointly to evaluate applicability of this more sophisticated approach.

References

[1] E. Hunsberger and C. Eliasmith. "Training Spiking Deep Networks for Neuromorphic Hardware". In: *CoRR* abs/1611.05141 (2016). arXiv: 1611.05141.

[2] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith. "Nengo: a Python tool for building large-scale functional brain models". In: *Frontiers in Neuroinformatics* 7 (2014), p. 48. ISSN: 1662-5196.

[3] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner. *TORCS, The Open Racing Car Simulator*. 2014. URL: http://www.torcs.org.

[4] M. R. Bonyadi, Z. Michalewicz, S. Nallaperuma, and F. Neumann. "Ahura: A Heuristic-Based Racer for the Open Racing Car Simulator". In: *IEEE Transactions on Computational Intelligence and AI in Games* 9.3 (2017-09), pp. 290–304.

[5] B. H. F. Macedo, G. F. P. Araujo, G. S. Silva, M. C. Crestani, Y. B. Galli, and G. N. Ramos. "Evolving Finite-State Machines Controllers for the Simulated Car Racing Championship". In: *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 2015-11, pp. 160–172.

[6] M. Ebner and T. Tiede. "Evolving driving controllers using Genetic Programming". In: *2009 IEEE Symposium on Computational Intelligence and Games*. 2009-09, pp. 279–286.

[7] L. Cardamone, D. Loiacono, and P. L. Lanzi. "Learning drivers for TORCS through imitation using supervised methods". In: *2009 IEEE Symposium on Computational Intelligence and Games*. 2009-09, pp. 148–155.

[8] E. Yee and J. Teo. "Evolutionary spiking neural networks as racing car controllers". In: *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*. IEEE, 2011-12, pp. 411–416. DOI: 10.1109/his.2011.6122141.

[9] E. Perot, M. Jaritz, M. Toromanoff, and R. d. Charette. "End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017-07, pp. 474–475.

[10] D. Rasmussen, A. Voelker, and C. Eliasmith. "A neural model of hierarchical reinforcement learning". In: *PLOS ONE* 12.7 (2017-07), pp. 1–39.

[11] F. Mirus, C. Axenie, T. C. Stewart, and J. Conradt. "Neuromorphic sensorimotor adaptation for robotic mobile manipulation: From sensing to behaviour". In: *Cognitive Systems Research* 50 (2018), pp. 52–66. ISSN: 1389-0417.

[12] D. Loiacono, A. Prete, P. L. Lanzi, and L. Cardamone. "Learning to overtake in TORCS using simple reinforcement learning". In: *IEEE Congress on Evolutionary Computation*. 2010-07, pp. 1–8.

[13] T. Bekolay, C. Kolbeck, and C. Eliasmith. "Simultaneous unsupervised and supervised learning of cognitive functions in biologically plausible spiking neural networks". In: *35th Annual Conference of the Cognitive Science Society*. 2013, pp. 169–174.

[14] C. Eliasmith and C. H. Anderson. *Neural Engineering : Computation, Representation, and Dynamics in Neurobiological Systems*. Computational neuroscience. Cambridge, Mass. MIT Press, 2003. ISBN: 0-262-05071-4.