A neuromorphic controller for a robotic vehicle equipped with a dynamic vision sensor

Hermann Blum*, Alexander Dietmüller*, Moritz Milde†, Jörg Conradt‡ Giacomo Indiveri†, and Yulia Sandamirskaya†
*Department of Information Technology and Electrical Engineering, D-ITET, ETH Zurich, Switzerland
‡Department of Electrical and Computer Engineering, TU Munich, Germany
†Institute of Neuroinformatics, University and ETH Zurich, Switzerland
Email: ysandamirskaya@ini.uzh.ch

***Abstract*—Neuromorphic electronic systems exhibit advantageous characteristics, in terms of low energy consumption and low response latency, which can be useful in robotic applications that require compact and low power embedded computing resources. However, these neuromorphic circuits still face significant limitations that make their usage challenging: these include low precision, variability of components, sensitivity to noise and temperature drifts, as well as the currently limited number of neurons and synapses that are typically emulated on a single chip. In this paper, we show how it is possible to achieve functional robot control strategies using a mixed signal analog/digital neuromorphic processor interfaced to a mobile robotic platform equipped with an event-based dynamic vision sensor. We provide a proof of concept implementation of obstacle avoidance and target acquisition using biologically plausible spiking neural networks directly emulated by the neuromorphic hardware. To our knowledge, this is the first demonstration of a working spike-based neuromorphic robotic controller in this type of hardware which illustrates the feasibility, as well as limitations, of this approach.**

I. Introduction

Collision avoidance is a key task for mobile robotic systems to ensure safety of both the robot itself as well as humans and objects in its environment. Navigation in an unknown environment in many robotic applications – rescue missions, space exploration, or work on remote construction sites – requires autonomy and optimised power consumption. Although current machine learning and computer vision systems allow autonomous navigation in real-world environments, power consumption of both the computing and sensory systems currently used in successful applications is enormous, draining the robot's power and generally taking away resources from other tasks.

Neuromorphic engineering aims to achieve efficient real-time low-power computation using principles of biological neural networks, implemented directly in hardware circuits [6, 13, 14]. These neuromorphic circuits feature massively parallel processing, co-location of computation and memory, and asynchronous data-driven (event-based) real-time computing. As these properties make real-time processing of large amounts of sensory information possible in an energy-efficient way, they are particularly interesting for autonomous robotic systems.

In terms of power consumption and area usage, analog circuit implementations of neural and synaptic dynamics are a very promising solution [21]. In large networks, this difference makes low-power on-board computation possible for tasks that would otherwise require power-hungry GPUs in more classical neural network implementations.

However, analog neuromorphic electronic circuits are known to be hard to control since their properties are sensitive to device mismatch and, e.g., thermal fluctuations [18]. We solve this problem by using a well-established neural-dynamic framework [23] that allows us to implement robust computing architectures on this hardware. Specifically, we implement a small neural architecture in neuromorphic hardware that controls an autonomous robotic system to perform reactive obstacle avoidance and target acquisition in an unknown environment. All computation for this system is done on the neuromorphic processor ROLLS (Reconfigurable On-Line Learning System) [22]. ROLLS is connected to the miniature computing platform Parallella, which is used to direct the real-time flow of spike events between the ROLLS and the robotic platform PushBot. Sensory input is provided by a Dynamic Vision Sensor (DVS) [12] and an inertia measurement unit of the robot. In this paper, we focus on verifying robustness of the developed architecture in different conditions and improving target representation by an allocentric memory mechanism.

II. Methods

A. Hardware

Fig. 1 shows the neuromorphic chip ROLLS on the Parallella board and the PushBot robot, used in this work.

1) ROLLS: The neuromorphic processor ROLLS is a mixed signal analog / digital neuromorphic platform [22]. The analog part includes 256 adaptive-exponential integrate-and-fire neurons. Each neuron exhibits biologically realistic neural behavior including a refractory period, spike frequency adaptation, and biologically plausible time constants of integration (e.g., tens of milliseconds). Connections between neurons – synapses – are also implemented using analog electronics and have biologically plausible activation profiles [8]. Each neuron has 256 programmable (non-plastic) synapses, 256 learning (plastic) synapses, which can be used to connect neurons to each other or to receive sensory signals, and 4 auxiliary ("virtual") synapses used to stimulate neurons directly.

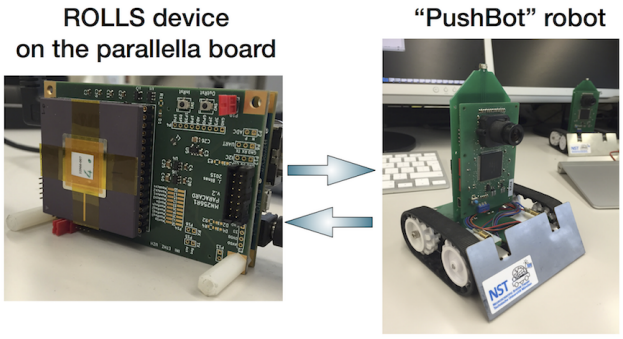


Fig. 1: The robotic setup used in this work: the neuromorphic processor ROLLS is interfaced wirelessly to the PushBot using a miniature computing board Parallella.

The programmable on-chip routing on the ROLLS that supports all-to-all connectivity allows us to implement arbitrary neural architectures. However, the plastic synapses can assume only one of 4 possible synaptic weight values that can be programmed via a 12-bit temperature compensated bias-generator.

2) PushBot and eDVS: The PushBot is a mobile platform with a differential drive. The robot is equipped with an inertial measurement unit (IMU), an LED at the top, and an embedded DVS silicon retina [17]. Each pixel of the 128x128 sensor array of the eDVS reacts asynchronously to a local change in luminance and sends out an event. Every event contains the coordinates of the sending pixel, the time of event occurrence, and its polarity ("on-event" or "off-event"). Due to the asynchronous sampling, the DVS is characterized by an extremely low latency, which results in μs time resolution, as well as low power consumption [11]. The embedded version of the DVS has an ARM Cortex microcontroller that initializes the DVS, captures events, sends them to the wireless network, and receives and processes commands for motor control of the robot.

The DVS produces a continuous stream of events in high-contrast areas, usually objects' boundaries, where changes are induced by sensor, or object motion. Additionally, we use the IMU sensor to get sensory feedback about the robot's heading direction (compass) and its angular velocity (gyroscope) and process signals from this sensor on the neuromorphic chip.

3) Parallella: The Parallella computing platform [19] is a 18-core credit-card sized computer, of which we only use one of its ARM-cores to run a simple software that configures ROLLS and integrates different parts of the hardware setup: it receives events from the eDVS and signals from the IMU, stimulates neurons on ROLLS according to the camera events and IMU signals, collects spikes from ROLLS, and sends drive commands to the robot. The only computation done on the Parallella is computing spike rates from different groups of silicon neurons and sending them as commands to the robot.

B. Neuronal architecture

The core of this work is a neuronal architecture that allows the robot to navigate in an unknown environment based on the output of its sensors (DVS and IMU). This neuronal architecture amounts to a connectivity matrix, set between the silicon neurons of the ROLLS chips that is shown in Fig. 3. Next, we describe different parts of this neuronal architecture.

1) Robot control: We model the desired PushBot movement a with forward velocity v and an angular velocity $\dot{\varphi}$. We encode both variables with the average firing rates of populations of neurons on the ROLLS. To encode the sign of $\dot{\varphi}$, we use two populations of equal size that inhibit each other and represent turning right and turning left, respectively. The decision of turn direction is "taken" in ROLLS, since only one of the turning populations can be active at the same time. The turning velocity is proportional to the average activity rate in the winning neuronal population. We use three populations of 16 neurons each to represent 'angular velocity (left)', 'angular velocity (right)', and 'speed' (forward velocity).

On Parallella, the firing rates are computed by counting the number of spikes from the respective neuronal population – $n_{\text{left}}, n_{\text{right}}$, and n_{speed} – in regular sampling intervals. These counts determine the velocities:

$$v \sim n_{\text{speed}}, \qquad \dot{\varphi} \sim n_{\text{left}} - n_{\text{right}},$$

normalized for the size of the neural population and the sampling time. To improve the reaction time, a first order low pass filter is implemented to update an estimate for spike rates:

$$n_{\text{estimate}} = \alpha \cdot n_{\text{old\_estimate}} + (1 - \alpha) \cdot n_{\text{count}},$$

where the desired time constant τ of the time-continuous low-pass filter and the sampling time T determine α as $\alpha = \exp\left(-\frac{T}{\tau}\right)$. We used a sampling time of 50ms and a time constant of 100ms resulting in $\alpha = 0.6$. The current firing rate per neuron and second, multiplied by a user-defined scaling factor, is sent to the robot (every 50ms).

2) Obstacle Avoidance: The first goal of our neural architecture is the reactive obstacle avoidance. We used a Braitenberg-vehicle principle [3] to realise obstacle avoidance based on the DVS output, which can also be cast as an attractor dynamics approach [1].

We only consider the lower half of the DVS field of view (FoV) for this task, since objects in the upper half are either above the robot or far away and therefore will not cause collisions. A population of 32 neurons on ROLLS represents obstacles. Columns of 4 $\times$ 64 DVS pixels are mapped to one neuron each. For every event in a column, the respective neuron is stimulated. After sufficient stimulation, the neuron will spike and therefore signal the detection of an obstacle.

The obstacle population is connected to the velocity populations described in section II-B1 (Robot control). The half of the obstacle population representing obstacles on the left/right have excitatory connections to the 'angular velocity (right/left)' population, respectively. Following the reactive architecture of a Braitenberg vehicle, the robot turns away in

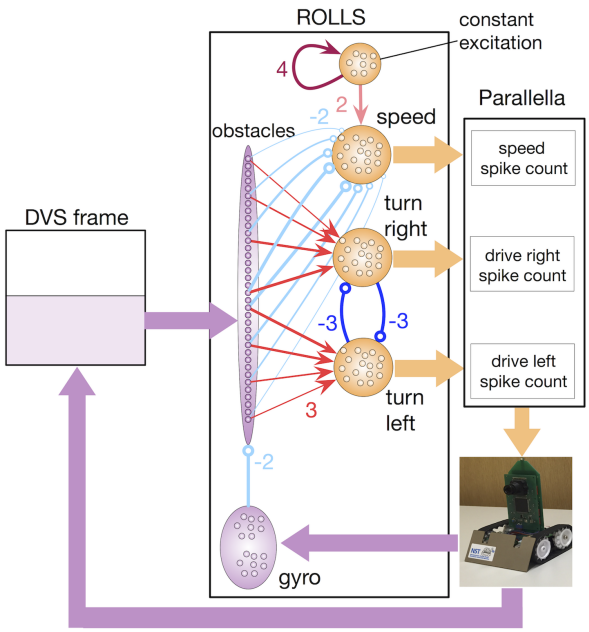


Fig. 2: Overview of the obstacle avoidance architecture, implemented on the neuromorphic chip ROLLS.

response to obstacles: if there is more input left than right, the robot turns right, otherwise left. Connections to the velocity populations from neurons representing obstacles in the center of FoV are stronger than for obstacles on the periphery (note the arrow thickness in Fig. 2). This makes obstacles avoidance smoother.

In the absence of obstacles, the robot drives straight forward. In order to represent the "default" speed for this case, we implement a constantly excited population of 8 neurons that excites the speed population. All neurons in the obstacle population, in their turn, have inhibitory connections to the speed population (Fig. 2), causing the robot to slow down in the presence of obstacles (with a stronger deceleration for bigger/more obstacles, which cause more DVS events).

Since the number of available weight values on ROLLS is limited (see section II-A1), we achieve the graded connections between neuronal populations by varying the number of connecting synapses to the respective population. Thus, neurons representing obstacles in the center of the FoV are connected to all 16 neurons in the velocity populations, whereas neurons representing obstacles on the pheriphery of the FoV are connected to only one (randomly selected) neuron in the velocity population; the number of connections decreases linearly when approaching the periphery of the FoV.

Because of the nature of the DVS camera, the robot detects more obstacle events when turning than when moving forward and the rate of the DVS events increases proportionally to the angular velocity. To compensate for this effect, we inhibit the obstacle detecting neurons while turning. This inhibition is realized using the gyroscope of PushBot. The implementation is described in more detail in section II-B4 (Proprioception).

In conclusion, we were able to implement obstacle avoidance using raw DVS input with just 88 artificial neurons by carefully grouping them and linking the different neuron groups in order to distinguish obstacle positions and react accordingly: strong reactions for obstacles in front of the robot, weaker reactions for more peripheral obstacles. Inhibition of the obstacle populations during robot turning was critical for robust obstacle avoidance, as well as slowing down in the presence of obstacles.

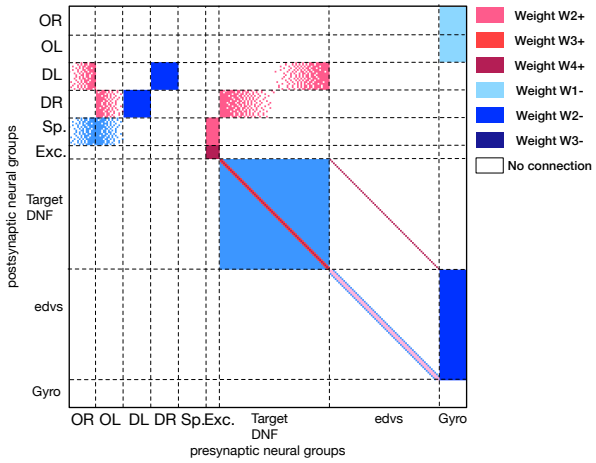


Fig. 3: Connectivity matrix of the full neural architecture with obstacle avoidance, target acquisition, and proprioception (input from gyroscope).

3) Target Acquisition: We simplified target perception in this work, because of the limited number of neurons on ROLLS in the current realization (256 neurons). More advanced architectures for target detection can be implemented in neuromorphic hardware [15], but were not the focus of this work. The target in our experiments is an LED of the second PushBot, blinking at 4kHz. The LED generates DVS events at a high rate, thus being a salient input even with a high number of distractors and sensor noise. Our goal is to detect this target and keep it in memory if the target vanishes for short periods of time. In particular, we would like to keep it in memory in allocentric coordinates, doing the coordinate transformation on the ROLLS chip, eventually.

We realized target acquisition with two populations of 64 neurons each. The first population is used as a filter for the DVS input. Similar to obstacle avoidance, every neuron in this population receives input from columns of 2×64 DVS pixels, this time from the upper half of the image. Since the neurons require a steady stream of events to emit spikes themselves, this population effectively filters out sensor noise. Additionally, neurons are connected with a local winner-takes-all (WTA) kernel amplifying local maxima of activity.

The second layer represents the target position (working memory for target position). Every neuron in the filter population excites exactly one neuron in the target memory population. In the target memory population, we use a global WTA dynamics: every neuron excites its close neighbors while

inhibiting all other neurons in the population. This connectivity 'selects' the global maximum out of the local maxima that the filter layer produces and also creates a "working memory" for the angular position of the target if it is lost from sight [25]. Similar mechanism has been used previously in the attractor dynamics approach to robot navigation [2].

The neurons in the target memory population are connected to the velocity populations: neurons representing a target on the left excite the 'turn to the left' population, and vice versa. To make the robot turn faster for the target on the periphery of the FoV than for a target in the center, neurons that represent targets in the center are connected to a single neuron in the turn population and neurons representing targets on the edge excite all neurons in the turn population. The number of connections linearly increases towards both edges of the target memory population.

Both target memory population and obstacle population are connected to the angular velocity populations. To ensure that obstacle avoidance is always prioritized over following the target, the connections from target acquisition are weaker than those from obstacle avoidance.

With this architecture, we enable the robot to follow the target and avoid an obstacle if necessary. While we can keep the target in memory, we are not able to adapt the 'remembered position' while the robot is turning, which can lead to undesired behavior (described in section III-F). One approach how this problem can be solved with neuronal populations is described in section II-B5.

4) Proprioception: As described above, we receive many more events from DVS while turning, which can lead to turning movements being longer than necessary, since the additional events are recognized as obstacles and keep the robot turning. Therefore, we need proprioception to recognize that the robot is turning and inhibit the neurons receiving DVS events as a countermeasure, similar to how saccadic suppression works in the mammalian eye [5].

We are using the gyroscope output to determine when the robot is turning. In contrast to DVS events, this sensor outputs integer numbers sampled every 50ms, which can not be directly used to stimulate ROLLS. Therefore, using the range of the sensor output, we transform the sensor output value into a number of spikes for stimulating ROLLS.

On ROLLS, we define two populations of eight neurons each to represent 'turning to the left' and 'turning to the right'. Every sampling step of the sensor, they receive the number of stimulations, proportional to the output value of the gyroscope.

Finally, we use these populations to inhibit all populations that receive input from the DVS, i.e. the obstacle and the DVS-filter populations. In this way, we successfully adjust sensitivity of the perceptual neural populations on ROLLS depending on the sensed turning to compensate for the additional events.

5) Extension of Target Memory: As the experiments in section III-F show, our first target acquisition architecture from section II-B3 fails if the target is out of sight. Since the target representation is stored in image-based coordinates, the memorized location of the target becomes invalid as the robot turns without updating the target memory.

To address this problem, we suggest a mechanism to store an absolute target position in memory instead of the relative (i.e. image-based) position, making use of the compass sensor and a neuronal architecture for reference frame transformations [24], realized in the ROLLS device.

This mechanism allows us to combine the absolute heading direction of the robot, which can be obtained from the compass of the IMU, with the relative target position found by processing DVS events, to obtain the absolute ("allocentric") angular position of the target with respect to a fixed rotational coordinate frame.

The target position in a world-fixed angular reference frame is updated as long as the target is in view and is held in a memory mode if the target is lost from view. The memorized position is transformed back into the image-based target representation through the same reference-transformation network and can be used to drive the robot back towards the target.

Using 108 neurons, we realized a version of this transformation on ROLLS that distinguishes 6 different heading directions (see Fig. 4). It was possible to tune this architecture such that the memory was updated as long as the target was in the FoV and the memory was kept if the target was lost. The transformation between coordinate frames is accomplished in a 6×6 matrix (see [24] for details of a continuous version of this mechanism), where 2 neurons on ROLLS represent each entry (cell) of the matrix. Only one matrix cell where heading (0-5) and memory (a-f) directions intersect is active at a time, "predicting" the relative position of the target (I-VI). If the target is detected with the camera, a strong input on the heading direction of the robot (I) overwrites the memory.

In Fig. 4, we show the mechanism at work. Here, the input from the IMU inhibits the target detection when the robot is turning; the representation in memory is used then to update the target position. The number of neurons on ROLLS did not allow us to implement this architecture together with DVS processing to filter out noise from cluttered backgrounds and could therefore not be used in our experiments.

6) Implementing the Architecture: Internally, our neural architecture amounts to a connectivity matrix. We developed a simple C++ library to fill-in this matrix, connecting neurons or neuron groups in various ways (e.g., all-to-all, winner-take-all, random, weighted). The software allows to define and connect neural populations, as well as to link them to inputs / outputs.

III. Experimental Results

The robustness of our obstacle avoidance setup as well as its limitations were tested in a wide range of experiments. We tested it against different types of obstacles and in different lighting conditions. All experiments were run for at least three times. However, the actual trajectories and neural activities differ between experiments too much to show a useful synthesis of different experimental runs, we thus show one of the runs for each experiment.

Most of our experiments are set in a controlled 'arena' environment with both white floor and walls. To make the

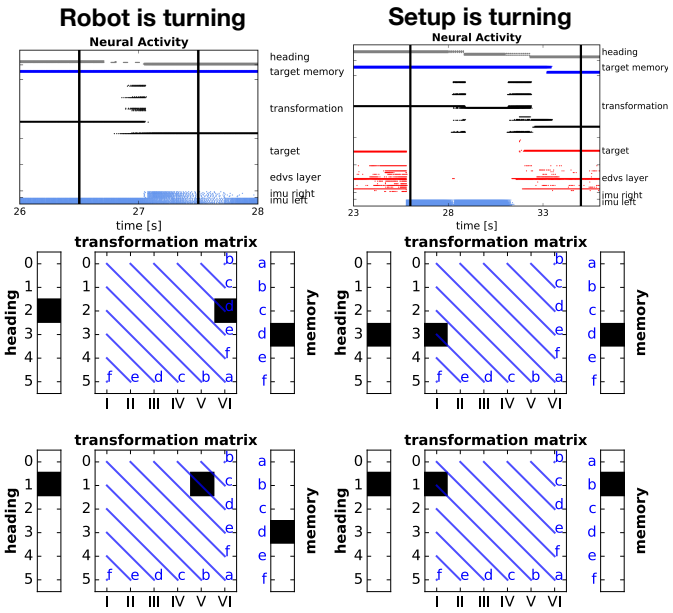


Fig. 4: Mechanism for transformation of the target position memory into an allocentric reference frame. **Top**: Spikes of neurons on the ROLLS chip in the neural populations, involved in transformation of the target position. **Middle and bottom**: transformation matrix and visualisation of its inputs (black squares) for the two points in time marked with the vertical lines in the respective Top plot (middle plots for the left line in the respective Top plot, bottom plots for the respective right line). Two experiments are shown here: **Left:** the robot is turning counterclockwise with a fixed target, the heading direction switches from position "2" to "1", but the memory of the (allocentric) direction to the target stays constant. **Right:** the whole setup with the robot and the target in front of it is turned on a platform; both the heading direction of the robot and the memorised allocentric direction towards the target shift.

walls visible to the DVS, we attached a high-contrast tape to the top of the walls. We had several runs in the office environment, which will be reported elsewhere. Next, we present a number of results that highlight properties of the architecture.

A. Different Obstacle Positions

Fig. 5 shows the robot's response to different obstacle positions. The robot starts with the same initial position and heading. The obstacle is an ordinary cup. Initially, it is placed directly on the robot's path and it is shifted to the right from the initial heading direction of the robot by 5cm per experiment.

The experiment qualitatively shows the expected difference in the magnitude of the robot's response. For an obstacle that is less in its way and therefore closer to the edge of the DVS's FoV, both the turn command and the slowdown are weaker.

We also observe that not only the position of the obstacle,

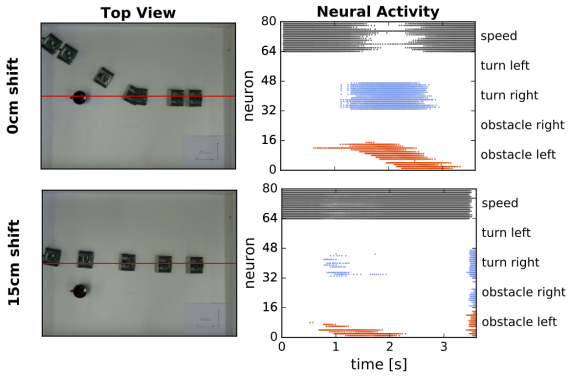


Fig. 5: Response of the robot to different obstacle positions. **Left**: Overlays of the overhead camera images at fixed time intervals. The red line marks the initial heading direction of the robot. When the obstacle is on the robot's pathway (top), it causes a stronger deviation from the initial trajectory than when the obstacle is positioned 15cm to the right from the line of the initial heading (bottom). **Right**: Activity of neurons on the ROLLS chip for the neural populations involved in generation of the avoidance maneuver.

but also the size of the obstacle on the DVS image is of importance. Activity of neurons in the top row of Fig. 5 shows that a single neuron in the obstacle population (red spikes) is not enough to excite the turn population (blue spikes). Only as the robot gets closer to the obstacle and therefore the obstacle occupies more columns of the DVS image and excites more obstacle neurons, the activity is strong enough to start a turn.

We can conclude that our architecture indeed leads to the intended weaker response to obstacles that are not directly in front of the robot. However, our setup will also avoid wider obstacles with a stronger response than small, narrow obstacles.

B. Different Colors

Different colors of obstacles lead to different contrasts to the background and thus to different amounts of DVS events as the edge of the obstacle moves in the FoV. Fig. 6 shows the behavior of the robot moving towards a black, red, and yellow obstacle (approx. 5cm height and 3cm diameter).

We observe that although the obstacle populations detect obstacles of all three colors, the distance to the obstacle at the time of the first spike decreases from the black to the red and to the yellow obstacle. Thus, with the ROLLS's biases used, the neural activation threshold is too high to avoid yellow obstacles. They provide a sufficient number of DVS events to activate the turn population only when the robot is already too close and their input is too weak to cause a turn strong enough to avoid the obstacle at this point.

Overall, we could find that PushBot in our setting reliably avoids obstacles of black, red, green, and blue color, while it regularly ignores yellow obstacles. Regardless of the bias setting, our principle of detecting an obstacle by rate of DVS events, and only using the filtering capabilities of spiking

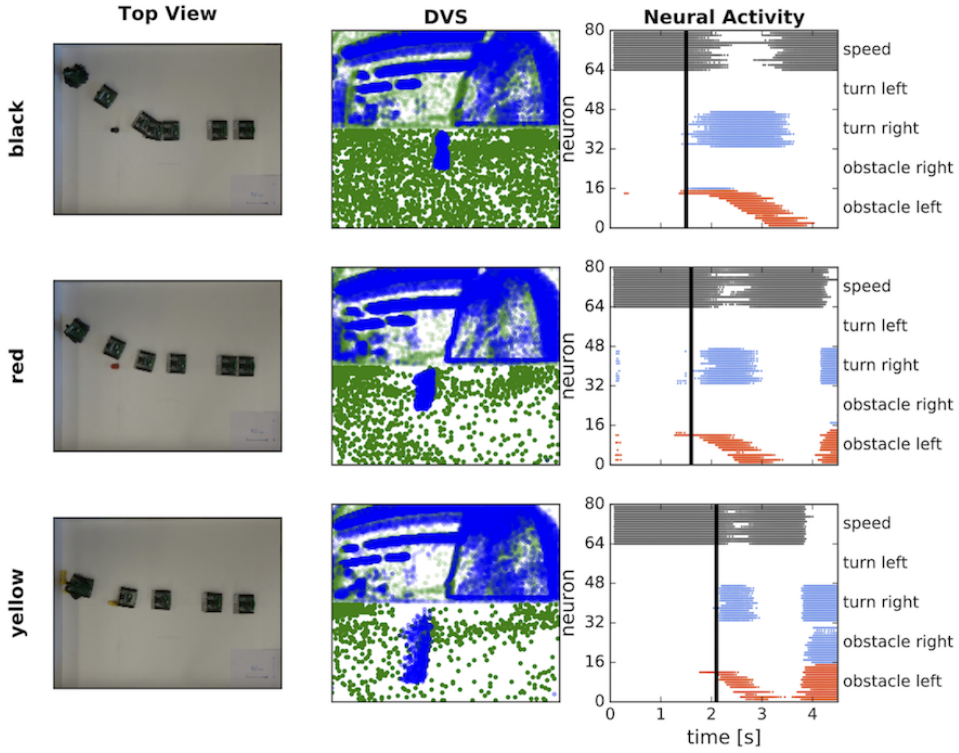


Fig. 6: Response of the robot to different obstacle colors. **Left**: Overlays of the overhead camera images with fixed time intervals to indicate the speed. **Center**: Image of DVS events accumulated over 1.5s up to the start of the turn, indicated by the black vertical line in the neural activity plot. **Right**: Neural activity on the ROLLS for the neural populations that control the avoidance maneuver.

neurons, requires to determine an arbitrary threshold that balances the robustness against noise vs. the sensitivity to low-contrast colors[1].

Additionally, we could show that reliable avoidance of yellow obstacles is also possible by changing the connection weights with which the obstacle population excites the turn and inhibits the speed populations, but this leads to the robot navigating slower (it decelerates stronger and more often) and turning stronger for obstacles with high contrast.

C. Different Lighting Conditions

In this set of experiments, the robot is placed in the same initial position for all experiment runs. The experiment was done in the evening, so there was no sunlight, and we used different office lights to simulate varying lighting conditions. Several obstacles were put on the robot's path to test the response to obstacles both when driving straight and while turning.

Fig. 7 shows the robot's trajectory for 2 different lighting conditions, both being darker than daytime experimental setup. They are representative examples for the general robot behavior at these lighting conditions as we tested each condition for at least 3 times. The obstacle does not get recognized below a certain level of brightness, as the contrast of obstacles in front of a background is obviously dependent on lighting conditions.

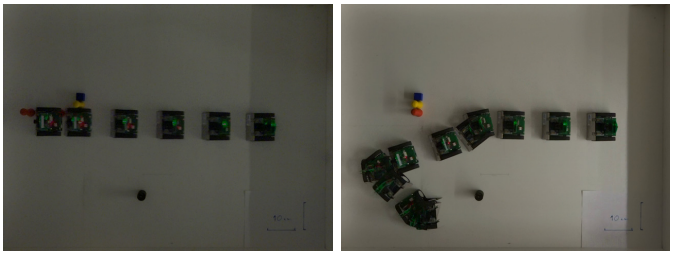


Fig. 7: Overlayed images of the robot trajectory in the arena at different lighting conditions. **Left**: Dark, robot fails to perceive an obstacle. **Right**: Lighter, but still less light than at day time, the robot avoids obstacles successfully.

D. Moving Obstacles

Moving obstacles are of special interest for implementations of obstacle avoidance as they are very common in real world navigation problems and require the ability to react to changing environments. The robot is placed in the same initial position for all experiment runs. Initially, there is no obstacle present in its FoV. After the robot starts moving

[1] This threshold can be changed by changing the ROLLS bias setting for the stimulating synapses, changing the number of synapses used for one stimulation, or changing the number of stimulations per DVS event.

forward, an obstacle is moved in its way. This procedure is repeated with different distances between the robot and the obstacle and different speeds of the obstacle. The robot is successfully avoiding the moving obstacle without difficulties, since a moving obstacle, in general, generates more DVS events than a static one.

E. Cluttered environment and the proprioceptive feedback

We show that our architecture enables the robot to navigate in a cluttered environment. The robot is placed in the arena that is populated with black cylinders, roughly 5cm high and 3cm in diameter, as obstacles. The cylinders are placed arbitrarily.

We find that the robot is able to avoid most obstacles 'on-the-go', i.e. without stopping, and is also able to drive through relatively narrow gaps (~ 1.5x the robot's width).

Fig. 8 shows the robot avoiding obstacles in a cluttered environment using proprioception (right) and without proprioception (left), which inhibits sensory events when the robot is turning. Comparing *Position 1* on both sides of Fig. 8, the greater activity in the obstacle population without gyroscope shows the lacking inhibition from the gyroscope populations. This leads to keeping the robot turning while it actually could pass between the two objects. Without inhibition from the gyroscope, the avoidance maneuver is much longer and the gap between the two cylinders in front of the robot (although big enough) is not used. In addition, the forward velocity of the robot is lower.

Nevertheless, the robot is able to navigate the cluttered environment without collisions with and without gyroscope, but we conclude that by using the gyroscope (proprioception) the robot is able to drive faster and go through narrower gaps while turning more smoothly.

F. Target Acquisition

The experiment was conducted with a static target that was an LED of the second PushBot blinking at 4 kHz with 75% on-time. The navigating PushBot was placed in the same position for all experiment runs with the target to the left of the initial heading direction. On the line between the two robots, we placed a small black obstacle.

Fig. 9 shows a snapshot of neural activity for one exemplary run of the experiment. The robot successfully approaches the target while avoiding the obstacle. The ROLLS activity shows that the single target is successfully detected and tracked by the WTA "target" population.

The shape of the robot's trajectory that can be seen in Fig. 10a is the result of an attractor-repellor dynamics between the target acquisition and obstacle avoidance. The number of connections from the target and obstacle representing layers to the turn populations depend on the position of the target, or obstacle in the FoV. Thus, the strength of the target attractor and of the obstacle repellor increase or decrease as the robot moves.

The main limitation we could find in these experiments is that the robot will loose the target if it has to turn away because of an obstacle (Fig. 10b). Even though the target representation on ROLLS has an "inert" (memory-like) behavior, the robot will not update the relative target position in memory as it turns. Keeping track of the absolute target position using architecture presented in II-B5 would allow the robot to turn back to the target that was lost from sight.

In addition to the presented experiments, we did successfully test target acquisition in the office environment. Furthermore, we did conduct tests where the target was not stable but moved around, remotely controlled by the experimenters. We did in general find that moving targets were followed as long as they did not move much faster than the autonomous robot and if they did not move outside of the FoV.

We could show here a working combination of target acquisition and obstacle avoidance, in which the decision of which direction to follow is taken by the competitive dynamics between the 'turn-left' and 'turn-right' neural populations on ROLLS. These populations receive inputs from the obstacle and target neurons, forming an attractor-repellor system.

In our current implementation, the robot speed had to be slow enough to detect the target (approx. 0.5 of the robot's maximal speed). This was necessary to reduce the events from the image background (due to the movement of the DVS) with respect to the signal from the blinking LED. Better noise filtering could allow faster movement.

IV. Discussion

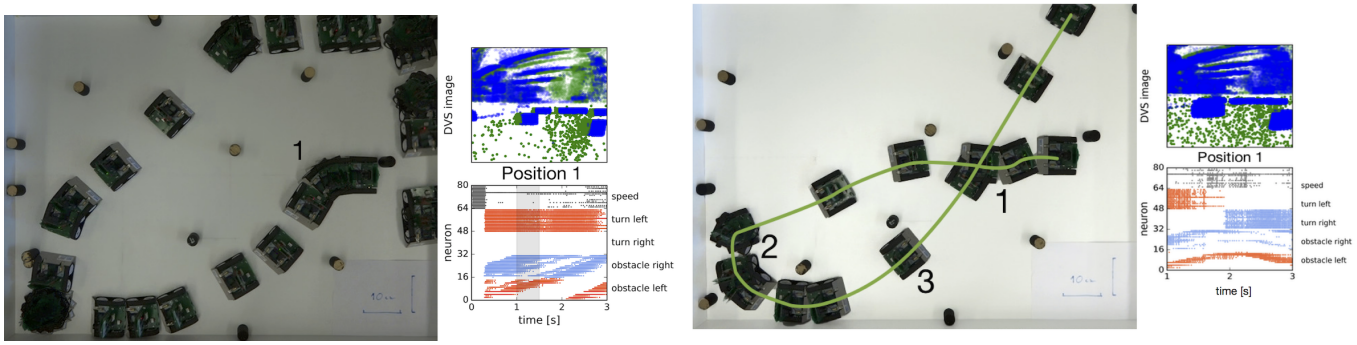
In this paper, we demonstrated that neuromorphic hardware can be used to implement both obstacle avoidance and target acquisition using only 256 spiking neurons. The robot is able to navigate cluttered environments, avoid moving obstacles, and follow a target at the same time. All the 'behavioral' decisions are made in real-time directly on the neuromorphic hardware.

When combining obstacle avoidance and target acquisition, the limited number of weights available on the hardware becomes a problem. Indeed, it was unavoidable to use the same weights in different parts of the architecture, leading to complex interference in the tuning process.

There are more limitations of the current system: we make use of all available neurons, making it impossible to extend our work with additional behaviors. Larger neuromorphic processors already exist [7] and will allow us to expand the repertoire of behaviors in our robot.

The number of neurons can not only be increased by building larger neuromorphic devices, but by connecting multiple devices. For the architecture described here, it is actually possible to separate the architecture in different modules: the neural populations for obstacle position and target position do not influence each other, they only receive inputs from the IMU and the DVS and output to the command populations. Therefore, in future work multiple ROLLS chips can be used to implement different architectural modules, resembling the classical subsumption architecture [4].

While our experiments show that obstacle avoidance and target acquisition can be achieved by processing the raw DVS events, this simple approach could be extended by introducing



(a) No proprioception: several collisions occur.

(b) Proprioception is used: all obstacles are avoided.

Fig. 8: **Left**: Overlayed overhead camera images with fixed time intervals. The marked with "1" point corresponds to the plots on the right. **Right**: Image of the DVS events accumulated over 0.5s at the indicated robot position (top). Neural activity on the ROLLS for the neural populations that control the robot's movement (bottom).

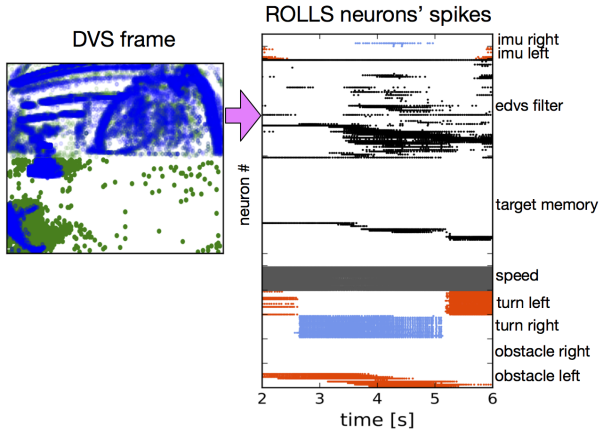
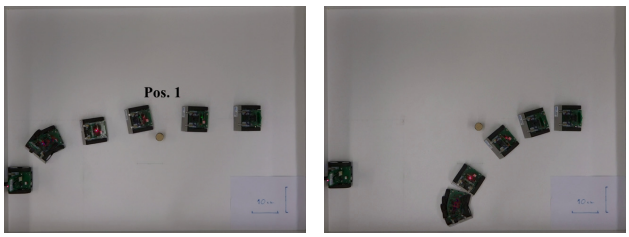


Fig. 9: Robot approaching a target robot while avoiding an obstacle on the way. **Top**: Image of DVS events accumulated over 1.5s marked by the gray area in the bottom plot. **Bottom**: Activity on the ROLLS for the labeled neural populations.



(a) Successful target acquisition and obstacle avoidance.

(b) Target is lost from sight after avoiding an obstacle.

Fig. 10: Limitations of the target acquisition in image-based reference frame.

preprocessing of the events stream. Possible solutions and extensions to the visual processing would require more neurons, and could rely on the recent progress in spiking neural networks [15, 16].

Since we introduced a way to compensate the limited number of weights in the ROLLS by varying the number of synaptic connections between neural populations, we consider the number of neurons a harder limitation than the number of weights.

PushBot platform has shown to be well-suited for our task, but it lacks the possibility to be directly connected to the neuromorphic processor. We have bridged this gap in software on Parallella, but for future implementations it will be advantageous to have a hardware interface that can be driven by spikes to provide a more direct link between the neuronal activity and the robot motion, as suggested in [20].

Overall, our proof of concept implementation is an important step, contributing to a growing field of neuromorphic controllers for robots [26, 9, 10, 16, 17], since we present a simple yet flexible architecture for spiking neuromorphic VLSI[2] devices that can easily be extended with additional functionality.

Acknowledgments

This was work financially supported by EU H2020-MSCA-IF-2015 grant 707373 ECogNet, "Forschungskredit" grant of the University of Zurich FK-16-106, and a fellowship of the Neuroscience Center Zurich.

References

[1] E Bicho, P Mallet, and G Schöner. Using attractor dynamics to control autonomous vehicle motion. In *Proceedings of IECON'98*, pages 1176–1181. IEEE Industrial Electronics Society, 1998.

[2] Estela Bicho, Pierre Mallet, and Gregor Schöner. Target representation on an autonomous vehicle with low-level sensors. *The International Journal of Robotics Research*, 19(5):424–447, 2000.

[3] V Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT press, 1986. ISBN 0262521121. doi: 10.2307/2185146.

[2]Very Large Scale Integration

[4] R A Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2: 12–23, 1986.

[5] David C. Burr, M. Concetta Morrone, and John Ross. Selective supression of the magnocellular visual pathway during saccadic eye movements. *Nature*, 371:511–513, 1994.

[6] E. Chicca, F. Stefanini, Ch. Bartolozzi, and G. Indivei. Neuromorphic Electronic Circuits for Building Autonomous Cognitive Systems. *Proceedigns of the IEEE*, 102(9):1367–1388, 2014.

[7] G. Indiveri, F. Corradi, and N. Qiao. Neuromorphic architectures for spiking deep neural networks. In *Electron Devices Meeting (IEDM), 2015 IEEE International*, pages 4.2.1–4.2.14. IEEE, Dec. 2015.

[8] Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii C Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud, Johannes Schemmel, Gert Cauwenberghs, John Arthur, Kai Hynna, Fopefolu Folowosele, Sylvain Saighi, Teresa Serrano-Gotarredona, Jayawan Wijekoon, Yingxue Wang, and Kwabena Boahen. Neuromorphic silicon neuron circuits. *Front Neurosci*, 5:73, 2011. ISSN 1662-453X. doi: 10.3389/fnins.2011.00073.

[9] Scott Koziol and Paul Hasler. Reconfigurable Analog VLSI circuits for robot path planning. *Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 36–43, 2011. doi: 10.1109/AHS.2011.5963964.

[10] Jeffrey L. Krichmar and Hiroaki Wagatsuma. *Neuromorphic and brain-based robots*, volume 233. 2011. ISBN 9781607509585. doi: 10.3233/978-1-60750-959-2-209.

[11] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128 X 128 120db 30mw asynchronous vision sensor that responds to relative intensity change. *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, pages 2004–2006, 2006. ISSN 0193-6530. doi: 10.1109/ISSCC.2006.1696265.

[12] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.

[13] C Mead. Neuromorphic Electronic Systems. *Proceedings of the IEEE*, 1990.

[14] Paul Merolla and Et Al. Artificial brains. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science (New York, N.Y.)*, 345(6197):668–73, 2014. ISSN 1095-9203. doi: 10.1126/science.1254642. URL http://www.ncbi.nlm.nih.gov/pubmed/25104385.

[15] S. Mitra, S. Fusi, and G. Indiveri. Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Transactions on Biomedical Circuits and Systems*, 3(1):32–42, 2009. ISSN 19324545. doi: 10.1109/TBCAS.2015.2479256.

[16] Diederik Paul Moeys, Federico Corradi, Emmett Kerr, Philip Vance, Gautham Das, Daniel Neil, Dermot Kerr, and Tobi Delbrück. Steering a Predator Robot using a Mixed Frame / Event-Driven Convolutional Neural Network Steering a Predator Robot using a Mixed Frame / Event-Driven Convolutional Neural Network. (July), 2016.

[17] Georg R. Müller and Jörg Conradt. A miniature low-power sensor system for real time 2D visual tracking of LED markers. *2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011*, pages 2429–2434, 2011.

[18] Emre Neftci, Elisabetta Chicca, Giacomo Indiveri, and Rodney Douglas. A systematic method for configuring VLSI networks of spiking neurons. *Neural computation*, 23(10):2457–2497, 2011. ISSN 0899-7667.

[19] Andreas Olofsson, Tomas Nordström, and Zain Ul-Abdin. Kickstarting high-performance energy-efficient manycore architectures with epiphany. In *2014 48th Asilomar Conference on Signals, Systems and Computers*, pages 1719–1726. IEEE, 2014.

[20] Fernando Perez-Peña, Arturo Morgado-Estevez, Alejandro Linares-Barranco, Angel Jimenez-Fernandez, Francisco Gomez-Rodriguez, Gabriel Jimenez-Moreno, and Juan Lopez-Coronado. Neuro-inspired spike-based motion: from dynamic vision sensor to robot motor open-loop control through spike-VITE. *Sensors (Basel, Switzerland)*, 13(11):15805–15832, 2013. ISSN 14248220. doi: 10.3390/s131115805.

[21] N. Qiao and G. Indiveri. Scaling mixed-signal neuromorphic processors to 28nm fd-soi technologies. In *Biomedical Circuits and Systems Conference, (BioCAS), 2016*, pages 552–555. IEEE, 2016.

[22] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in neuroscience*, 9:141, 2015.

[23] Yulia Sandamirskaya. Dynamic Neural Fields as a Step Towards Cognitive Neuromorphic Architectures. *Frontiers in Neuroscience*, 7:276, 2013.

[24] Sebastian Schneegans and Gregor Schöner. A neural mechanism for coordinate transformation predicts presaccadic remapping. *Biological cybernetics*, 106(2):89–109, 2012.

[25] G Schöner. *Dynamical systems approaches to cognition*. Cambridge University Press, 2008.

[26] Terrence C. Stewart, Ashley Kleinhans, Andrew Mundy, and Jörg Conradt. Serendipitous Offline Learning in a Neuromorphic Robot. *Frontiers in Neurorobotics*, 10 (February):1–11, 2016. ISSN 1662-5218. doi: 10.3389/fnbot.2016.00001.