2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Error estimation and correction in a spiking neural network for map formation in neuromorphic hardware

Raphaela Kreiser[1], Gabriel Waibel[1], Nuria Armengol[1], Alpha Renner[1], Yulia Sandamirskaya[1]

Abstract— **Neuromorphic hardware offers computing platforms for the efficient implementation of spiking neural networks (SNNs) that can be used for robot control. Here, we present such an SNN on a neuromorphic chip that solves a number of tasks related to simultaneous localization and mapping (SLAM): forming a map of an unknown environment and, at the same time, estimating the robot's pose. In particular, we present an SNN mechanism to detect and estimate errors when the robot revisits a known landmark and updates both the map and the path integration speed to reduce the error. The whole system is fully realized in a neuromorphic device, showing the feasibility of a purely SNN-based SLAM, which could be efficiently implemented in a small form-factor neuromorphic chip.**

I. INTRODUCTION

Animals with even relatively small neural systems –e.g., insects– can orient themselves in an environment, find paths to their goals, and form associations between places and landmarks [1], [2], [3]. We might wonder how they solve all the computing tasks involved in foraging with sometimes as few as some 100K neurons. Neurorobotics helps to understand neuronal circuits that control the behavior of animals by using such circuits to control robots, thus testing the neuronal models in a closed behavioral loop [4]. Neuronal simulations typically require powerful computers to run in real-time, making them not well-suited for real-time control. Neuromorphic engineering has brought about neuronally inspired computing hardware that allows us to implement biologically realistic Spiking Neuronal Networks (SNNs) on computing chips with high efficiency in real time [5], [6], [7], [8], [9], [10], making them promising for robotic applications.

Conventional software and controllers are not suitable for "programming" neuromorphic hardware. Instead, the whole processing pipeline must be developed using neuronal networks. While end-to-end learning of a neuronal controller is one way to find an architecture that solves a particular task, in this work, we use an alternative approach. We design the neuronal architecture following inspiration from known circuits in animal brains that evolved to solve similar problems. Learning and adaptation are reserved for task-related adjustments to the architecture. In particular, we use on-chip plasticity (local learning rules) to learn locations of landmarks in a given environment and to adjust parameters of path integration to match the movement of the robot.

[1]Raphaela Kreiser, Gabriel Waibel, Nuria Armengol, Alpha Renner, and Yulia Sandamirskaya are with the Institute of Neuroinformatics, University of Zurich and ETH Zurich, 8057 Zurich, Switzerland rakrei@ini.uzh.ch; ysandamirskaya@ini.uzh.ch

Other authors have started working towards neuromorphic SLAM and have recently shown how an SNN can form a 1D-map on a neuromorphic chip that can be used to navigate to the learned locations [11]. These efforts follow the first successful realization of a bio-inspired RatSLAM architecture that was used to map a real office environment [12].

The contribution presented here builds on our previous work implementing a number of components of neuromorphic Simultaneous Localization and Mapping (SLAM) using SNNs on chip [13], [14], [15]. We have previously realized 1D (heading direction) and 2D (position) estimation networks for a wheeled robotic vehicle, based on path integration of motor commands and visual cues [13], [14]. In a second step, we presented a simple map formation with the map being updated at loop closure events, but without corrections to the path integration process [14]. Finally, we have introduced a mechanism to autonomously adapt the rate of path integration upon error detection [15], [16]. In this paper, we integrate the developed components of the neuromorphic SLAM –path integration, map formation, and correction of both the path integration speed and the map– for a one-dimensional case.

In our experiments, the robot rotates on a spot and detects visual cues placed around it. When the same visual cue is revisited, loop closure is detected, and the location of the cue, stored in memory in the SNN, is compared to its location, estimated based on path integration. If a mismatch is detected, the error magnitude is estimated by the network and is used to correct the potential error sources (either the map or path integration network). In previous work, this task of map correction at loop closure was solved outside the neuronal SLAM architecture [12], [17]. The proposed here neuronal mechanisms for mismatch detection and error estimation are fully realized on the neuromorphic chip.

We have implemented each architectural component with spiking integrate-and-fire neurons on Intel's neuromorphic research test chip Loihi [5]. The network is tested both with simulated and real robotic data, showing online learning, forgetting, and adaptation. This work shows the feasibility of a fully neuronal SLAM architecture that runs in neuromorphic hardware, leading to an efficient and adaptive system with power consumption below 1W.

II. MATERIALS AND METHODS

A. Neuromorphic Hardware: Loihi

An overview of spiking neuromorphic processors can be found in [18]. In this work, we realized the SNN architecture on Intel's neuromorphic research test chip Loihi [5], in

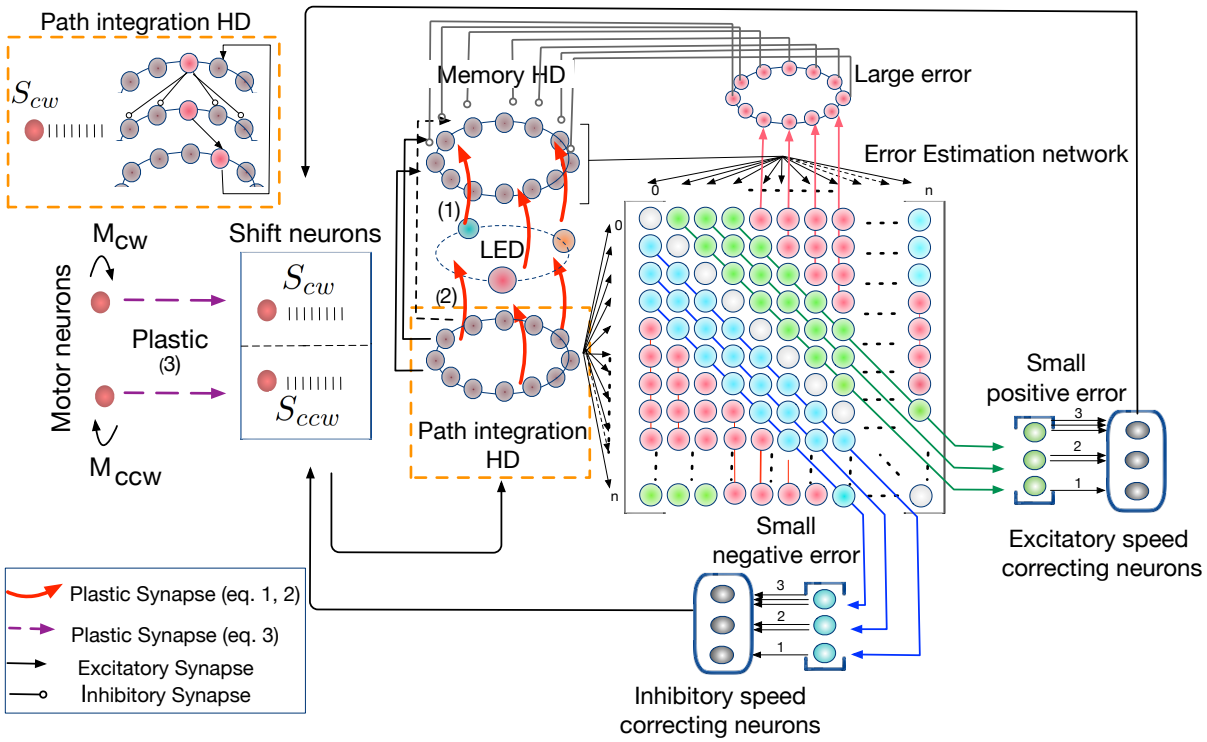

Fig. 1. Overview of the SNN SLAM architecture: (1) Two ring attractor networks encode the robot's heading direction (HD) based on the neuronal path integration and memory of previously detected visual cues (blinking LEDs); (2) Shift neurons drive the active neuron in the path integration HD network to move with a speed determined by their firing rate; (3) At a loop closure event, the error estimation network (2D array of neurons) estimates the magnitude and sign of the difference between the location of the active neuron in the memory-driven HD network and the path integration HD network; (4) Small errors drive Shift neurons and trigger plasticity in the synapses connecting them to the motor neurons. Large errors induce map updates. See main text for details.

particular, its small form-factor version Kapoho Bay. Loihi uses an asynchronous digital design to implement event-driven parallel computations. The processor also implements on-line learning with local plasticity rules. The chip consists of 128 cores realizing up to 128K artificial neurons and features a unique programmable microcode learning engine for on-chip SNN learning.

B. SNN architecture

Fig. 1 illustrates the schematics of the network architecture that solves the following tasks: (1) 1D orientation estimation, (2) map formation, (3) error estimation, and (4) error correction. We describe these parts of the architecture next:

1) HD network: The orientation estimation network consists of a population of neurons that form a ring attractor network and perform 1D path integration. We call this population the "path integration Heading Direction (HD)" population. Each neuron in the path integration HD population represents an angular range, the whole ring spanning 360° around the robot. At any given time, at most, one neuron is active (emits a spike) in the HD population and represents the robot's current orientation in a global coordinate frame.

The neural activity in the HD population is shifted when the robot turns by the following mechanism. Two neuronal populations –the S_{cw} and S_{ccw} neurons– activate shift layers that move the neural activity in a clockwise (CW) or counter-clockwise (CCW) direction, respectively (see inset of Fig. 1, upper left corner). The speed of the movement of neuronal activation depends on the firing rate of the S_{cw} or S_{ccw} neurons. The path integration HD is connected to another ring-population (shift layer) that is activated by either S_{cw} and S_{ccw} neurons that shift activity to the left or to the right, respectively. The shift layers feed their activation back to the HD population. Due to input integration in neurons (Eq. 1), when the spike frequency of input neurons (S_{cw} and S_{ccw}) is higher, neurons in the HD population require a shorter amount of time to cross the activation threshold. This leads to faster transfer of the activation in the network and, thus, a faster movement of activity in the HD population, i.e. a higher neural path integration speed. The dependence of the represented movement speed on the firing rate of shift neurons is shown in Fig. 2. See [13], [15] for more details of the HD network architecture.

$$\dot{u}_i(t) = -\frac{1}{\tau_u}u_i + \sum_j w_{ij}\cdot\delta_j(t), \quad \dot{v}_i(t) = -\frac{1}{\tau_v}v_i(t) + u_i(t). \quad (1)$$

Eq. 1 describes how spike trains ($\sum_j \delta_j(t)$) are scaled with the synaptic weight w_{ij} and integrated in a low-pass filter process to obtain the input current $u_i(t)$. This synaptic input, in its turn, is integrated in the neuron's membrane potential $v_i(t)$. A fixed threshold parameter determines the amount of synaptic input needed to elicit a spike in the neuron, after which $v_i(t)$ is reset to zero, and the integration continues.

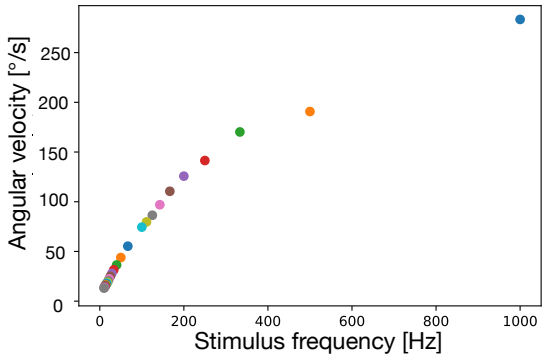

Fig. 2. The firing rate of shift neurons ("Stimulus frequency") determines how fast the activity in the HD network moves and thus the speed of neuronal path integration ("Angular velocity").

The S_{cw} and S_{ccw} neurons receive excitatory input from motor neurons M_{cw} and M_{ccw}, respectively, that are activated when the robot receives a command to move in a CW or CCW direction. The motor neurons are connected to shift neurons with plastic synapses. The learning rule in these synapses changes the firing rate of the S_{cw} and S_{ccw} neurons in response to spikes of the respective motor neuron. This firing rate controls the speed of activity movement in the HD population, allowing us to match it to the actual movement of the robot when errors are detected in a loop closure event.

2) Map formation: The path integration HD network estimates the robot's orientation based on motor commands. At the same time, visual cues are detected in the environment and are used to form a 1D map of the environment as the robot rotates on the spot. The robotic setup used in our experiments is shown in Fig. 3. We place LEDs around the robot with different blinking frequencies to side-step complex visual processing. The LEDs are sensed with a dynamic vision sensor (DVS) camera [19], and an event-based vision routine detects different blinking frequencies in the center of the DVS's field of view. When an LED is detected, one of the LED neurons is activated using spike generators on Loihi.

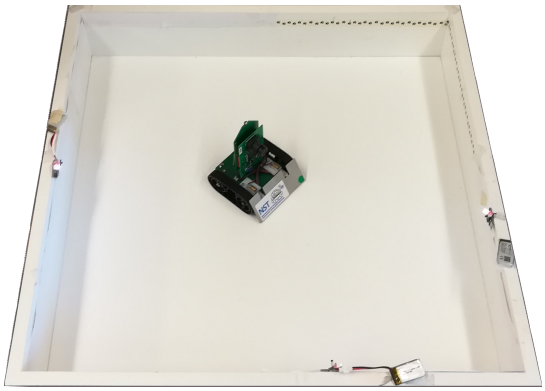

Fig. 3. The experimental setup: a Pushbot robot rotates on a spot, surrounded by blinking LEDs. Positions of the LEDs are learned in an SNN and updated, along with the path integration speed, at loop closure events.

The path integration HD population is connected to LED neurons with plastic synapses that are initialized with zero weights. These plastic synapses are strengthened or weakened according to a spike-time dependent local learning rule, Eq. 2. After learning, the activity in the path integration HD population causes recall of the LED that has been previously detected at the respective position.

To enable recall of the robot's orientation based on the detected visual cues (LEDs), we introduce another ring attractor network – the memory HD network shown in Fig. 1 above the LED neurons. The memory HD population receives excitatory one-to-one input from the path integration heading direction (HD) neurons. Plastic synapses leading from LED neurons to memory HD form associations between visual landmarks and the robot's orientation when a blinking LED is detected by the robot. These plastic synapses are also initialized with zero weights and updated according to the learning rule Eq. 3:

$$\Delta w_{HD\to LED} = y_0 \cdot x_1 \cdot (y_1 - c_1) - x_0 \cdot (x_1 + c_2), \quad (2)$$

$$\Delta w_{LED\to Mem} = x_0 \cdot y_1 \cdot x_1 - y_0 \cdot (y_1 + c_2), \quad (3)$$

where $w_{HD\to LED}$ and $w_{LED\to Mem}$ are weights of the synapses connecting path integration HD neurons to LED neurons and LED neurons to memory HD neurons, respectively; x_0 and y_0 are binary variables that turn from 0 to 1 when the pre- or the postsynaptic neuron spikes and turn back to 0 in the next time step; x_1 and y_1 are pre- and postsynaptic traces computed as spikes convolved with an exponentially decaying temporal kernel; c_1 and c_2 are constants balancing synaptic potentiation (weight increase) and depression (weight decrease).

During learning, the plastic synapses from path integration HD neurons to LED neurons follow Eq. 2. During their co-activation at a postsynaptic spike ($y_0 = 1$), the weight increases (the first term in Eq. 2) if postsynaptic activity trace is above a threshold c_1. The synaptic weight decreases when the presynaptic neuron, i.e., the path integration HD neuron, is active while the postsynaptic neuron, i.e., the LED neuron, is inactive (the second term in Eq. 2).

The weights of the synapses from LED neurons to the memory HD increase if both the pre- and postsynaptic neurons are active at a presynaptic spike (both presynaptic trace x_1 and postsynaptic trace y_1 in Eq. 3 are high). If a memory neuron (postsynaptic) is active while the LED neuron (presynaptic) is silent (no LED is detected), the weight decreases, and the previously learned association is unlearned.

Thus, the path integration HD neurons learn to expect an LED/landmark. The memory HD neurons represent the heading direction inferred both based on a previously learned position of the currently perceived LED/landmark and the path integration. The two inputs may overlap when no error has accumulated since the last encounter of the landmark, or not, in which case an error is detected and its magnitude is estimated.

3) Error estimation: When an already learned LED is detected in a loop-closure event, a mismatch (error) in the HD populations can be detected. We distinguish two error classes: small errors that probably arise from a path integration offset and large errors that are probably caused by changes in the environment. Small errors should reset the orientation estimate to the one inferred based on the learned map and update the path integration speed depending on the sign of the error. Large errors indicate a change in the environment and facilitate forgetting of the previously learned LED-associations and learning of the currently observed ones.

The central part of the error-correcting circuit is the error estimation network (a 2D neuronal array in Fig. 1). This network consists of a 2D layer of neurons that receive input from the path integration HD population and from the memory HD neurons. These two 1D inputs enter the 2D network along different dimensions, and their spikes are integrated by the neurons in the error estimation network. The neuron, for which the two active neuron arrays overlap, is activated.

Neurons in the error estimation network are connected to an output layer that consists of two read-out populations, representing the positive or the negative error. The position of the active neuron in one of the read-out populations corresponds to the difference between the positions of the active neuron in the path integration HD and the memory HD populations and thus represents the magnitude of the error.

4) Error correction: Apart from the error magnitude, the error estimation network detects the sign of the error, which signals if the orientation estimated through path integration is ahead or behind the orientation that is inferred from the detected LED and the activated memory HD neuron.

Neurons marked green and blue in Fig. 1 correspond to a small error that triggers calibration of the path integration network. Two groups of neurons representing a small negative error (blue) and a small positive error (green) read out the sign and magnitude of the estimated error. To convert the error into firing rates of the Shift neurons, these groups are connected to two pools of speed correcting neurons. These neurons translate the space-code of the error read-out neurons (the identity of the active neuron represents the error magnitude) to rate code, where the firing rate of the neural population represents the encoded value. To achieve this, each neuron in the positive or negative error population is connected to a certain number of the speed correcting neurons, proportional to the detected error magnitude (Fig. 1): e.g., the error neuron that represents the lowest error is connected to a single speed-correcting neuron, while an error neuron that represents a high error is connected to a dozen of speed correcting neurons.

The shift neurons (S_{cw} and S_{ccw}) are driven by plastic synapses from the motor neurons (M_{cw} and M_{ccw}), which send an efference copy of the motor command to the path integration system. The two speed correcting populations decrease or increase the firing rate of the shift neurons: positive error neurons are excitatory and increase the firing rate, while negative error neurons are inhibitory and decrease the firing rate. This temporary correction is then made permanent by inducing a weight change in the plastic synapses from M_{cw} and M_{ccw} to S_{cw} and S_{ccw} according to the following error-modulated Hebbian learning rule:

$$\Delta w_{M\to S} = (x_0 \cdot y_1 + y_0 \cdot x_1) \cdot r \cdot (w_{max} - w_{M\to S}) - x_0 \cdot r. \quad (4)$$

Here, r denotes the reinforcement (error) trace that is set to 1 by the detected error and decays exponentially over time. The maximal weight is limited by a value of $w_{max} = 120$.

Whenever the postsynaptic firing rate is higher than the presynaptic one, the synaptic weight is more likely to increase: the weight is increased if the pre- and postsynaptic neurons are active at the same time and the weight is decreased upon every presynaptic spike, not followed or preceded by a postsynaptic spike. The negative weight update brings about the property that higher pre- than postsynaptic firing rate leads to synaptic weight depression. Thus, the weight increase or decrease is controlled by the firing rate of the postsynaptic (shift) neurons during error detection.

When the orientation estimation in the path integration HD matches the LED-induced orientation in memory HD, the error is zero, and the learned synaptic weights determine the firing rate of the S_{cw} and S_{ccw} neurons and with that the path integration speed, i.e., the movement speed of the active neuron in the path integration HD.

C. Neuromorphic robotic setup

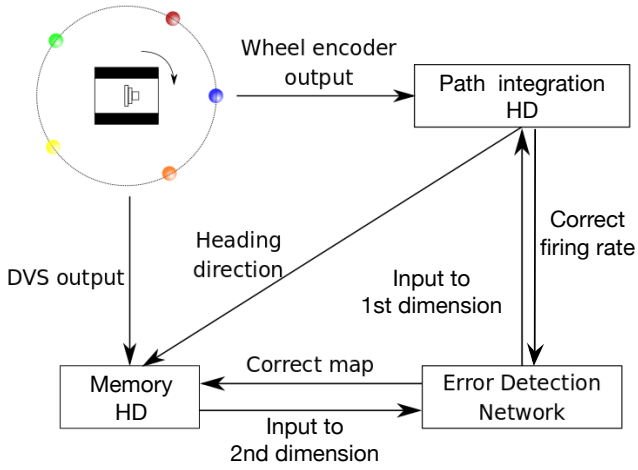

Fig. 4. Activation flow in the neuromorphic architecture: Motor signals are sent to the path integration HD network, while visual information from DVS is sent to the LED layer, connected to the memory HD network. The memory HD also receives input from the path integration HD. The memory HD builds the map and drives the error detection network, which corrects the map and path integration speed.

We evaluate the system's performance with simulated and real robotic data in a changing environment. The robot used in this work is a mobile platform called Pushbot, which consists of a 10×10cm chassis with two motors driving two independent tracks for propulsion. It comprises an embedded

DVS – a neuromorphic event-based camera [20], [21]. Each pixel of the DVS is sensitive to the temporal change in luminance and sends out an event using the Address Event Representation (AER) protocol. The differential values of the robot's wheel encoders are used to generate input spike trains with frequencies proportional to the robot's angular velocity. The initial frequency is chosen arbitrarily and is adjusted by the network through learning when loop closures are detected.

Based on the DVS output, we detect LEDs with three different frequencies that are used as landmarks. The angular positions of the LEDs are learned in synapses from LED to the memory HD neurons. Mismatches between the learned orientation and the currently estimated one trigger error detection. Fig. 4 shows an overview of the interaction between the robot and the neural network architecture. Fig. 3 shows the robot in the arena with three blinking LEDs placed on the walls.

III. Results

A. Learning and forgetting of visual landmarks

To show how the network in Fig. 1 learns associations between the estimated heading directions and LEDs, we simulate a scenario with two LEDs placed in an environment while the robot is rotating (see Fig. 5).

The HD networks in this experiment have 10 neurons. The top plot in Fig. 5 shows the neural activity of the path integration HD neurons. The middle and bottom plots show the synaptic weights connecting the path integration HD neurons to LED neurons and the LED neurons to the memory HD neurons, respectively. Both spikes and weights are colored according to the associated LED.

In this run, after 20 seconds, LED0 is removed, and the network unlearns the formed association. The position of LED1 (blue trace) happens to fall between two neighboring HD neurons and weights to two HD neurons are strengthened, to a lower value than weights to and from

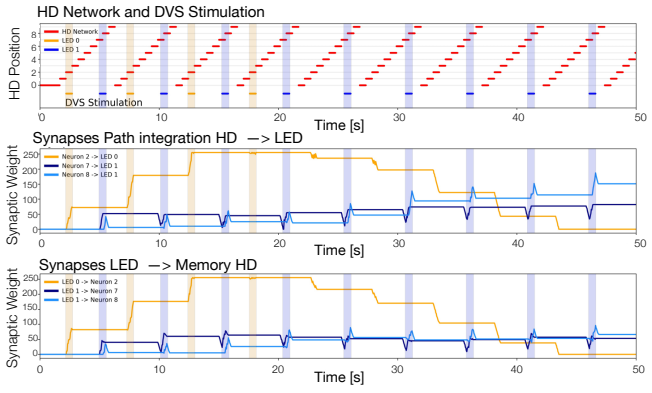

Fig. 5. **Top**: Neural activity of the path integration HD neurons while detecting two different LEDs (shown in blue and yellow). LED0 is removed after 20s. LED1 is placed between heading directions 7 and 8. **Middle**: Synaptic weights of the three HD neurons that learned the LEDs. **Bottom**: Synaptic weights from LED neurons to memory HD.

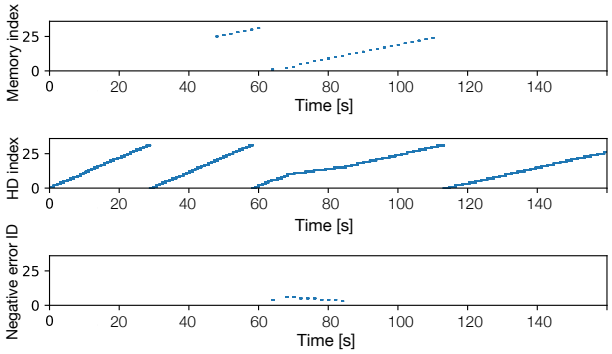

Fig. 6. Detecting a known landmark enables the network to adjust the path integration speed to match the turning speed of the robot. Here, the path integration (middle row) is too fast at first but gets corrected after detecting a negative error (bottom) when the true heading direction (top) is provided.

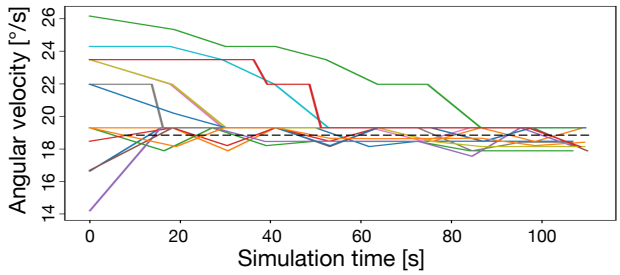

Fig. 7. Convergence to true angular velocity (black dashed line) starting with 14 different initial path integration speeds, determined by the input rates to the HD network. Simulation corresponds to 10 full rotations with a single landmark.

LED0. When LED0 (yellow trace) is removed, the respective weights decrease in every turn when the LED is expected, but not observed.

Next, we validate learning and forgetting using recorded robot data from seven consecutive turns of the robot in an environment with 3 different LEDs. We measure how many consecutive turns are necessary to achieve an effective synaptic weight, i.e., a weight that is strong enough to initiate postsynaptic activity. We found that most synapses are sufficiently strong after the first rotation, with a probability of 0.67. A small number of weights achieve a sufficient weight only after 4 full rotations. Most synapses require only one turn with a missing LED to decrease their synaptic weight.

B. Adaptation of the path integration speed

In the next experiment, we show how speed of the neuronal path integration can be adapted in the network. We simulate a scenario in which a set of landmarks are detected for a full turn. Fig. 6 shows neuronal activity recordings from the Loihi chip when landmarks are recognized from 47 to 110 seconds, and the memory HD population is continuously stimulated to represent this information. A set of negative error detecting neurons becomes active and changes the shifting speed of the neuronal activity in the HD population to match the true angular velocity of the robot, noticeable in the changing slope of the neuronal activity.

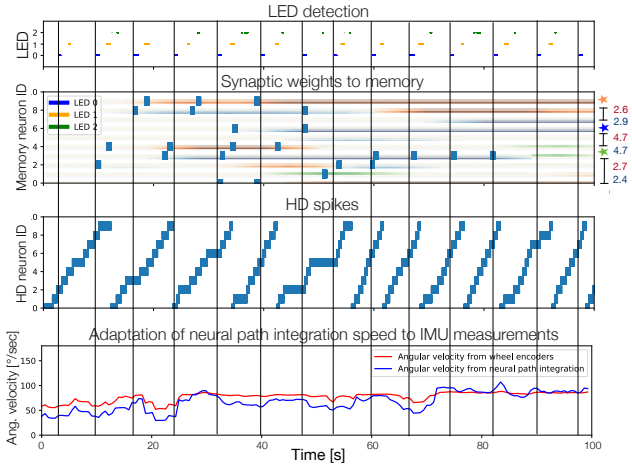

Fig. 8. From top to bottom: (1) Activity of three LED neurons detecting LEDs blinking at different frequencies. (2) Synaptic weight values from LED to memory HD neurons. Colors denote the identity of the associated LED. Stars indicate the true positions of LEDs. The true angular distances between the LEDs with respect to the robot and converted to neuron units are annotated in blue while the learned angular distances are shown in red. Blue rectangles denote "large error" neuron's spikes. (3) Activity of the path integration HD neurons. (4) Robot's angular velocity inferred from the neural path integration (blue) and obtained from wheel encoders (red). Note how mismatch is reduced after all errors are corrected.

To evaluate the system's learning performance over several encounters with the same landmark, we recorded activity during 10 simulated turns of the robot observing a single landmark. Landmark detection activates the corresponding neuron in the memory HD population, leading to error estimation. The path integration network slowly adjusts its integration speed by decreasing the weights of plastic synapses (Fig. 7).

Fig. 7 shows how the angular velocities, determined from the network's activity, converge over time to the true value of the robot's turning speed. Different colors correspond to trials with different initial path integration speeds in HD network. The black dashed line corresponds to the true angular velocity. In all trials, starting with different speeds, the neural network converges to match the speed of the moving activity to the true angular velocity of the robot.

C. Closed-loop robotic experiment

Finally, we examined the interplay of map formation and path integration calibration in a robotic experiment, creating a closed-loop setup with the Loihi chip and the Pushbot robot. The Pushbot is rotating in a small square arena with 3 LEDs placed on the walls that blink with different frequencies, see Fig. 3 and Fig. 4.

Fig. 8 shows the outcome of the experiment. The top panel shows the activity of the three LED neurons, the second panel shows the synaptic weights that are formed during detection. Small blue squares correspond to spikes emitted from neurons in the "large error" population, which induce forgetting of the previously learned LEDs. The third panel shows the neural activity in the path integration HD population, and the bottom panel shows angular velocity obtained from wheel encoders and from neural path integration.

It becomes apparent that up to second 50 the estimated and visually inferred heading directions do not match, as many error neurons are active (second plot in Fig. 8), leading to learning and forgetting of different associations between LED neurons and memory HD neurons. In the bottom panel, one can see that the neural angular velocity is slower than the one obtained from wheel encoders. From second 60 to 82, the neural path integration speed comes closer to the robot's angular velocity, and errors are detected only once in a while. Finally, the path integration speed equals the robot's angular velocity, and no errors are detected. The positions of the strong synaptic weights between LED neurons and memory HD neurons are close to true positions of the LEDs, indicated with stars in the second plot in Fig. 8. Thus, despite the very coarse resolution in the HD networks and unreliable detection of the blinking LEDs (upper plot), the system is capable to both calibrate its path integration speed and update positions of landmarks to correct values.

IV. CONCLUSION

We developed an SNN architecture for loop closure detection in a 1D SLAM scenario on the neuromorphic research chip Loihi. The neuronal architecture autonomously learns to match the internal representation of the robot's angular velocity to its actual turning speed and simultaneously creates a map of landmarks. The neuronal path integration is realized as the speed of movement of neural activity in a ring attractor network. Learning is gated by the detected mismatch between landmark locations recalled bottom-up from the path integration and top-down from memory of locations of visual cues.

The network structure is inspired by biological findings on navigational cell types in rodents and the head direction network of the fruit fly [3], [22]. This work presents a first architecture for loop closure detection and autonomous, online learning for calibration of a path integration system and map formation in an SNN, realized on neuromorphic hardware. Despite its proof of concept character, this step is essential to motivate scaling up of neuromorphic hardware systems.

ACKNOWLEDGMENT

This work was funded by SNSF Grant PZOOP2_168183 (Ambizione), Forschungskredit of the University of Zurich, grant no. [FK-76204-03-01] and ZNZ Fellowship. We would also like to thank Intel Labs for their support.

REFERENCES

[1] T. S. Collett and M. Collett, "Memory use in insect visual navigation," Nature Reviews Neuroscience, vol. 3, no. 7, p. 542, 2002.

[2] R. Menzel, R. Brandt, A. Gumbert, B. Komischke, and J. Kunze, "Two spatial memories for honeybee navigation," Proceedings of the Royal Society of London. Series B: Biological Sciences, vol. 267, no. 1447, pp. 961–968, 2000.

[3] W. E. Skaggs, J. J. Knierim, H. S. Kudrimoti, and B. L. McNaughton, "A model of the neural basis of the rat's sense of direction," in Advances in neural information processing systems, 1995, pp. 173–180.

[4] E. Falotico, L. Vannucci, A. Ambrosano, U. Albanese, S. Ulbrich, J. C. V. Tieck, G. Hinkel, J. Kaiser, I. Peric, O. Denninger, N. Cauli, M. Kirtay, A. Roennau, G. Klinker, A. Von Arnim, L. Guyot, D. Peppicelli, P. Mactinaz-Cañada, E. Ros, P. Maier, S. Weber, M. Huber, D. Plecher, F. Röhrbein, S. Deser, A. Roitberg, P. Van Der Smagt, R. Dillman, P. Levi, C. Laschi, A. C. Knoll, and M. O. Gewaltig, "Connecting artificial brains to robots in a comprehensive simulation framework: The neurorobotics platform," Frontiers in Neurorobotics, vol. 11, no. JAN, 2017.

[5] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," IEEE Micro, 2018.

[6] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," Frontiers in Neuroscience, vol. 9, no. APR, pp. 1–17, 2015.

[7] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A Scalable Multicore Architecture with Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," IEEE Transactions on Biomedical Circuits and Systems, 2018.

[8] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," Science, 2014.

[9] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker System Architecture," IEEE Transactions on Computers, vol. 62, no. 12, pp. 2454–2467, 2012.

[10] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems, 2010, pp. 1947–1950. [Online]. Available: http://web1.kip.uni-heidelberg.de/Veroeffentlichungen/download.php/4833/ps/2018.pdf http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5536970

[11] G. Tang, A. Shah, and K. P. Michmizos, "Spiking neural network on neuromorphic hardware for energy-efficient unidimensional slam," arXiv preprint arXiv:1903.02504, 2019.

[12] M. J. Milford, G. F. Wyeth, and D. Prasser, "RatSLAM: A Hippocampal Model for Simultaneous Localization and Mapping," Proceeding of the 2004 IEEE international Conference on Robotics & Automation, pp. 403–408, 2004.

[13] R. Kreiser, M. Cartiglia, J. N. Martel, J. Conradt, and Y. Sandamirskaya, "A Neuromorphic Approach to Path Integration: A Head-Direction Spiking Neural Network with Vision-driven Reset," in 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018.

[14] R. Kreiser, P. Pienroj, A. Renner, and Y. Sandamirskaya, "Pose Estimation and Map Formation with Spiking Neural Networks: towards Neuromorphic SLAM," in IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2018.

[15] R. Kreiser, A. Renner, and Y. Sandamirskaya, "Error-driven learning for self-calibration in a neuromorphic path integration system," in Robust AI for Neurororobotics Workshop, Edinburgh, 2019.

[16] R. Kreiser, G. Waibel, A. Renner, and Y. Sandamirskaya, "Self-calibration and learning on chip: towards neuromorphic robots," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Breaking news, 2019.

[17] O. Struckmeier, K. Tiwari, M. J. Pearson, and V. Kyrki, "ViTa-SLAM: A Bio-inspired Visuo-Tactile SLAM for Navigation while Interacting with Aliased Environments," arXiv preprint arXiv:1906.06422, 2019.

[18] C. S. T. Thakur, J. Molin, G. Cauwenberghs, G. Indiveri, K. Kumar, N. Qiao, J. Schemmel, R. M. Wang, E. Chicca, J. Olson Hasler, et al., "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," Frontiers in neuroscience, vol. 12, p. 891, 2018.

[19] J. Conradt, R. Berner, M. Cook, and T. Delbruck, "An embedded AER dynamic vision sensor for low-latency pole balancing," 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009, pp. 780–785, 2009.

[20] 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, pp. 2004–2006, 2006.

[21] S. C. Liu and T. Delbruck, "Neuromorphic sensory systems," pp. 288–295, 2010.

[22] S. S. Kim, H. Rouault, S. Druckmann, and V. Jayaraman, "Ring attractor dynamics in the drosophila central brain," Science, vol. 356, no. 6340, pp. 849–853, 2017.