Event-driven Vision and Control for UAVs on a Neuromorphic Chip

Antonio Vitale[1], Alpha Renner[2], Celine Nauer[2], Davide Scaramuzza[3], and Yulia Sandamirskaya[1]

Abstract— **Event-based vision enables ultra-low latency visual feedback and low power consumption, which are key requirements for high-speed control of unmanned aerial vehicles. Event-based cameras produce a sparse stream of events that can be processed more efficiently and with a lower latency than images coming from conventional cameras, enabling ultra-fast vision-driven control. Here, we explore how an event-based vision algorithm can be integrated with a spiking neural network based controller. When both are implemented on the same neuromorphic chip, seamless integration of perception and motor control can be demonstrated, as well as efficient online adaptation of the controller. Our spiking neural network on chip is the first example of a vision-based fully neuromorphic controller solving a high-speed control task. The excellent scalability of processing in neuromorphic hardware enables solving more challenging control tasks in the future, such as navigation in cluttered environments, vision-driven swarm robotics, or high-speed inspection and monitoring of infrastructure. Video of the experiments: https://youtu.be/uGEK8eT9Qb0**

I. INTRODUCTION

Autonomous unmanned aerial vehicles (UAVs) can potentially solve many tasks, e.g., autonomous infrastructure monitoring for predictive maintenance, supporting search and rescue operations, or delivering goods to remote areas. To be deployed in dynamic and unpredictable real-world environments, UAVs require fast and efficient perception and control [1]–[4]. It has been shown that event-based cameras—visual sensors with asynchronous pixels reporting local luminance change, inspired by biological retinas [5], [6]—lead to up to three orders of magnitude faster visual processing than conventional image-based vision sensors, while also consuming a few mW of power [4], [7]. Recently, it has been demonstrated how an event-based camera can enable tracking of a simple visual pattern with a constrained UAV at an angular velocity of $>$1500 degrees per second—performance, unreachable with image-based sensors and onboard computing [8].

Event-based cameras, such as the Dynamic Vision Sensor (DVS) [9], [10], generate a stream of events from their pixels instead of conventional image-matrices. Neither conventional computer vision algorithms nor convolutional neural networks (CNNs), typically used for advanced visual perception today, are directly applicable to this type of signals [7]. Thus, special event-based vision algorithms have been developed to extract task-relevant information from the stream of events [11]–[14] while for neural-network processing, event-images are usually created by accumulating events in conventional matrix structures based on fixed time intervals or a fixed number of events [15]–[18]. Special purpose accelerators have been proposed to make the processing of such DNN efficient for UAV applications [19].

Neuromorphic hardware originates from the same line of research as DVS and realizes brain-inspired computing principles on-chip such as: neuronal-network based computing, asynchronous event-based communication of neuronal activation with “spikes”, temporal dynamics of the state of neurons, and efficient on-chip learning in form of local synaptic plasticity. Neuromorphic computing devices—such as Intel’s neuromorphic research chip Loihi [20] used in this work and other systems [21], [22]—support event-based computation directly and enable low-power spiking neural network architectures with on-chip learning.

Spiking Neural Networks (SNNs) have been previously explored for vision tasks required for UAV control [23], using end-to-end learning and CNN to SNN conversion. Here, we show how an SNN for solving a visual line-tracking task can be implemented directly on neuromorphic hardware and can be connected to a PD controller, implemented in an SNN on the same neuromorphic chip. We compare the performance of the neuromorphic controller with the state of the art event-based vision-based controller, described in [8].

In our previous work, we have introduced both SNN-based proportional (P) controllers as well as proportional, integral, derivative (PID) controllers, all implemented in neuromorphic hardware [24]–[26]. This work inspired a number of recent neuromorphic motor control architectures [27] and continued the long-standing research line of neuronally inspired motor control methods [28], [29].

In this paper, we bring this work two steps further towards low-power vision-driven UAV control in real-world tasks: (1) we demonstrate how adaptation of the PID gains can be realised using on-chip learning rules; (2) we integrate the controller with a visual SNN directly driven by DVS-events. In contrast to previous work on ultra-fast event-driven motor control [8], [30], [31], here the DVS events are input directly to the neuronal cores of a neuromorphic chip, using address event representation (AER) interface [32], and processed directly by the SNN. This enables future research into more complex SNNs for efficient on-chip perception to solve different tasks required for autonomous navigation and mission control: target tracking and obstacle avoidance, visual odometry, and map formation. We benchmark different parts of the SNN architecture and the overall prototype system and conclude that the fully neuromorphic perception and control can bring out another order of magnitude improve-

[1]Antonio Vitale and Yulia Sandamirskaya are with Neuromorphic Computing Lab of Intel, Intel Labs, Munich, Germany.

[2]Alpha Renner and Celine Nauer are with Institute of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland.

[2]Davide Scaramuzza is with the Robotic Perception Group, at both Institutes of Informatics, University of Zurich, and Neuroinformatics, University of Zurich and ETH Zurich, Switzerland.

ment in speed to power-consumption trade-off compared to conventional control driven by event-based vision.

II. METHODS

A. Hardware

1) Neuromorphic hardware (Loihi): In this work, Intel's neuromorphic research chip Loihi [20] in the form of a Kapoho Bay device was interfaced with the rotating dual-copter. Kapoho Bay features two Loihi chips with a total of 256 neuro-cores in which a total of 262,144 neurons and up to 260 million synapses can be implemented. Three embedded x86 processors are used for monitoring and I/O spike management.

2) Constrained UAV: In this work, the same robotic platform was used as in [8] - a constrained to 1 DoF dual-copter that can rotate 360 degrees in front of an actuated visual pattern. A 3D-printed holder was designed to mount the Kapoho Bay on top of the rotating dual-copter. Fig. 1 shows both the mounted Kapoho Bay as well as a side view of the setup.

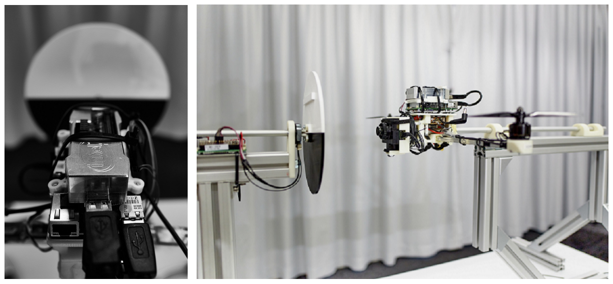


Fig. 1: **Left**: Intel's Kapoho Bay mounted on top of the dual-copter. **Right**: Side view of the entire physical setup.

3) Event Camera: The dual-copter is equipped with a DAVIS 240C event camera with a spatial resolution of 240x180 pixels and a temporal resolution of the asynchronous event stream of 1μs [9]. As in the previous work, the DAVIS sends the event stream to the UP board[1] via USB 2.0. According to [33], the duration of the vision processing, i.e., the elapsed time between an intensity change to the detection of the event by the on-board computer, is <5ms.

To explore the possibilities of on-chip state estimation and control, we connect the DAVIS camera to the Kapoho Bay via a direct AER [32] interface. The entire visual processing is done on the Loihi chip in this case, and its output is directly fed into the SNN PD controller running on the same chip.

4) ESCs and Motors: As in the original setup, the UP board sends motor commands to a Lumenier F4 AIO flight controller using the universal asynchronous receiver/transmitter (UART) communication protocol with a maximum transmission rate of 4.1kHz. The custom firmware for the flight controller was used that enables communication with the motors' electronic speed control (ESC) via the DShot150 protocol at a maximum rate of 9.4kHz. The ESC sends pulse-width modulation (PWM) signals to the motors, where the input voltage to the motors is given by the applied voltage (12.1V) times the duty cycle of the PWM signal.

The controller returns the required rotor thrusts to achieve the desired roll angle and these thrust values are converted to input voltages for the motors using a pre-calibrated thrust-to-duty-cycle mapping.

5) Rotary Encoders: The dual-copter and disk are both equipped with CUI AMT22 Modular Absolute Encoders that measure the ground truth roll angle and velocity. The encoders have an accuracy of 0.2° and a precision of 0.1°. The encoder transfers angle measurements to the UP board via Serial Peripheral Interface (SPI) with a maximum data transmission rate of almost 20kHz. In this project, we used an update rate of 1kHz.

6) Interfaces, system overview: The neuromorphic device, Kapoho Bay, was embedded on an UP board featuring an Intel® ATOM^{TM} x5-Z8350 Processor with 64 bits up to 1.92GHz and 4GB of RAM. The UP board is running Ubuntu 18.04, and the communication between the hardware was achieved with ROS Kinetic. Python 3.5.2 and version 0.9.9 of the Intel NxSDK were used to set up the SNN on the Loihi chip.

The state variable used for the control loop is the difference in orientation between the drone and the line on the disk. Here, we used two ways to obtain this measure: 1) As in the previous work, an event-based algorithm running on the UP board processes the event output stream to obtain the state estimation; 2) In order to exploit the advantages brought by the integration of the neuromorphic sensor with the neuromorphic processor, we directly connect the DAVIS camera to the Kapoho Bay and run the visual processing on-chip.

B. The Spiking Neural Network (SNN) PD Controller

The SNN used to control the drone's angular velocity of the drone in this work is based on our previous work [25] and [26]. Compared to the previous iterations, the on-chip controller SNN could be simplified here: the two inputs to the controller are directly provided by the visual module as the error in angle and its derivative. Also, in order to compare the software controller and the Loihi controller used on this platform, only a PD controller was used, thus allowing to omit the whole integration pathway introduced in [25]. Instead, we introduce an adaptive term in the controller that modifies the controller output in case the PD controller fails to remove the error (e.g., due to added load).

While in our previous work, we used a dual-copter moving at slower velocities and constrained to angles $\in (-25°, 25°)$, in addition to the faster computation time resulting from a better integration, two major improvements were vital in order to obtain good performance on this faster platform: 1) More efficient allocation of neuron compartments on the neuro-cores allowed for an increased population resolution and thus more precise control and a larger range of commands: the neuron populations' sizes were increased from

[1]Featuring Intel® ATOM^{TM} x5-Z8350 Processor with 64 bits up to 1.92GHz and 4GB of RAM

$N = 63$ to $N = 361$. 2) The SNN output was obtained at a higher rate via a faster output communication channel between the neuromorphic processor and the host computer (UP board) using direct memory access instead of channel communication over the embedded CPU: the readout time decreased from $\approx 1.5ms$ to $\approx 0.05ms$.

Fig. 2 illustrates the simplified Spiking Neural Network controller.

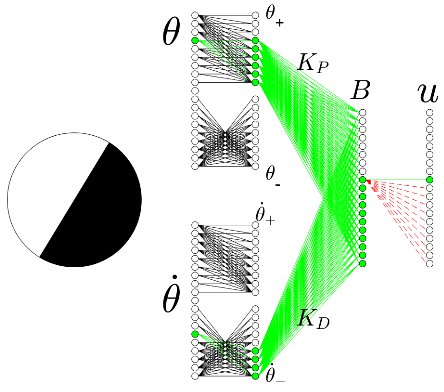


Fig. 2: The SNN controller network: The input layer consists of two populations, one representing the measured angular offset between the target orientation and the drone's roll angle, the second one representing the respective velocity offset. The output of these populations is scaled by the weights that represent the K_p and K_d gains and is summed in the auxiliary B population before it is projected on the control-output population u, in which only one neuron can spike at any time step. Green lines are excitatory, and red dashed lines – inhibitory synapses from the currently active neurons (green circles).

1) SNN Input/Output Encoding/Decoding: In order to send spikes to the Loihi chip, the host CPU (UP Board) communicates with one of the embedded processors of the Kapoho Bay using communication channels programmed using Intel's neuromorphic API NxSDK. To obtain the controller output, the output spikes are sent to a FIFO file on the host computer.

The input to the SNN in Fig. 2 in a first set of experiments is the output of the angle estimate in software. In this case, the values for the angle θ, under which the tracked horizon line appears in the DVS, and its rate of change, $\dot\theta$, are input to Loihi follwoing the place-coding mechanism: the range of possible values for each variable is divided into equal intervals and a neuron in the input layer is assigned to each interval. The neurons then receive an input space from the on-chip spike generator, depending on the measured value. When using the SNN Hough transform on Loihi, the network in Fig. 2 receives input directly as spikes on neuronal cores. The Hough SNN, in its turn, recieves input from the DAVIS camera, over the AER interface, as described later.

The spikes in the output population of the controller, u, encode the motor commands, also following the position encoding. The index of the firing neuron, $idx \in [0, 360]$, is decoded into the thrust difference according to Eq. 1. Here, $T_{min}, T_{max} = \pm 1850$, and N is the number of neurons in the output population:

$$\mathbf{u} = \frac{idx}{N} \cdot (T_{max} - T_{min}) + T_{min}. \quad (1)$$

The controller output is further converted in software to a thrust difference according to the equations:

$$\begin{aligned} \mathbf{u} &= K_P \cdot \theta + K_D \cdot \dot\theta, \\ \mathbf{T} &= c_T \pm (\frac{u}{2} + b_T), \end{aligned} \quad (2)$$

where T is the thrust; the constant $c_T = 0.4 \cdot T_{baseline} = 2880$ ensures that the motors are always returning a baseline thrust. The thrust bias $b_T = 170$ is an empirically chosen value; $u \in [-3700, 3700]$ is the controller output.

2) Adaptation Pathway: On-chip learning on Loihi enables adaption of the neuromorphic controller by using synaptic plasticity rules. Adaptive CPU controllers rely on monitoring the plant behavior and modifying the controller parameters accordingly. On the Loihi chip, adaptation is realized by exploiting the plastic connections between compartments (neurons) directly on the processor, only minimally increasing the processing overhead.

To test this capability, we introduced a constant, static disturbance to the plant, i.e. added a weight to one side of the drone. To compensate for this a-priori uknown disturbance, we use an additive feed-forward path that connects the input population, θ to the output, u. A similar approach was previously proposed for neuromorphic adaptive control of a robotic arm [28].

The adaptive path should compensate for extended periods of large steady-state errors. Thus, it needs to monitor the error signal over time. If the accumulated error exceeds a threshold, then the feed-forward term should increase or decrease appropriately. The adaptive mechanism is regulated by the integral in Eq. (3). The monitoring period, ΔT, and the cumulative error threshold, ϵ, are parameters which are empirically selected:

$$\frac{1}{\Delta T} \int_t^{t+\Delta T} err^2(t)dt < \epsilon. \quad (3)$$

In the network, we need to compute the integral in Eq. 3 using positional coding in the SNN. Fig. 3 illustrates the equivalent SNN-pipeline: the neuronal population that represents the error signal, $err(t) = \Delta\theta$, is connected to two neurons, one accumulating positive errors, R_+, and one – negative errors, R_-. These neurons have the activation thresholds and decay constants such that they only fire if the accumulated error reaches a threshold value. Each neuron in the $\Delta\theta$ (error) population is connected to the R_+/R_- neurons with a weight proportional to the error magnitude that the respective neuron represents: neurons closer to the center of the population, representing small errors, have a small weight and neurons on the periphery of the population (top and bottom in Fig. 3) are connected with a large weight.

The R-neurons are connected to two intermediate neurons, FF_+, FF_-, which are respectively connected with positive and negative weights to the population B and are thus responsible for increasing and decreasing the feed-forward term. The population B uses a special form of position encoding: depending on the represented value (of the integrated error), more or less neurons are active in this population. To achieve this, neurons in B have different activation thresholds (ordered from bottom to top in Fig. 3).

Learning on Loihi is realised by local synaptic plasticity rules that can be programmed by the user. The learning rule can be composed by a combination of pre- and post-synaptic spikes, auxiliary variables representing recent activity, and a third factor, called the reinforcement channel. The R-neurons are connected to the reinforcement channel of the synapses between the FF_+/FF_- neurons and the B population in such a way that when the R neurons fire, the synaptic weight changes according to the learning rule $\Delta W = W \pm \delta_R(t)$ where $\delta_R(t) = 1$ if the corresponding R neuron is firing, $\delta_R(t) = 0$ otherwise.

Fig. 3 shows the schematic of the adaptation pathway which runs in parallel to the full SNN PD controller shown in Fig. 2.

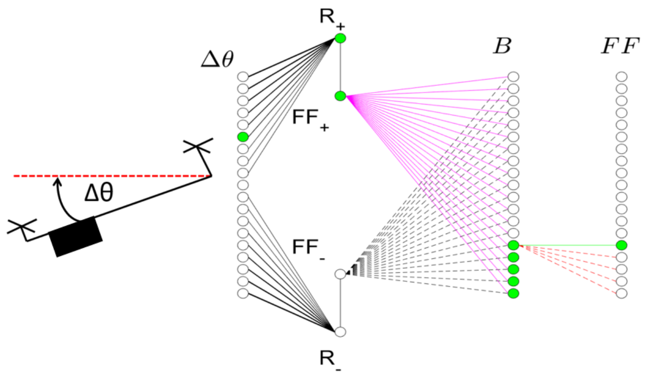


Fig. 3: Schematic of SNN Adaptation Pathway: Purple lines show currently active plastic synapses.

3) Vision Module: The Hough transform is a geometry-based computer vision algorithm for line detection in images [34], [35]. To apply the Hough transform, a line is parametrized by r and θ. Points laying on the same line in the Cartesian space map to intersecting sine curves in the Hough space (the θ-r space). Therefore, for many points on the same line, the (θ, r) pair with the highest intersection density represents the correct line in Cartesian space. In [8], an efficient version of the Hough transform for event-based input was used that updated an estimate of the Hough parameters of the line based on accumulated events in sliding 3ms intervals. For smooth state estimation, a Kalman filter was used.

Here, we propose a neuromorphic version of the Hough transform, similar to previous computational work [36]. The incoming DVS events are integrated in a 2D array of neurons representing (θ, r). The incoming pixels at location (x, y) are mapped to all pairs of (θ, r) that parametrize a possible line intersecting the coordinate (x, y). This effectively corresponds to doing the Hough transform by application of a single binary weight matrix to the incoming spikes. Due to the leaky integration of the neurons in the Hough layer, binning of spikes into event-frames is not necessary.

The binary connection matrix is computed according equation $r = x \cdot \cos\theta + y \cdot \sin\theta$. An excitatory synapse connects each pair of (θ, r) and (x, y), for which the equation holds. The continuous values for r are mapped on equally spaced bins of 10 pixels for $r \in (-200, 200)$, and the angles θ were mapped in the range $\theta \in (-90°, 90°)$ with a spacing of $2°$.

The full SNN for the angular error estimation is illustrated in Fig. 4. In a), the event-stream generated by the DVS camera mounted on the drone pointing at a black-and-white pattern is downsampled by 4 using an efficient 2-bit shift operation on the x86 co-processor on Loihi and then sent to a first layer b) of neurons on the Loihi's neuro-cores. In c), these neurons are connected to the Hough space neurons according to the generated connectivity pattern.

In d), the 2D (θ, r) space is read out in θ direction by connecting all neurons that correspond to the same θ to a respective readout ("angle") neuron, as seen in layer d). These angle neurons, in their turn, are connected to the second population e) of angle neurons that has the purpose of cleaning up activity in case several neurons are firing at the same time. To improve speed (sacrificing accuracy compared to recurrent winner-take-all connectivity that could be used in this layer instead), this is achieved by inhibiting all neurons with a lower index in a feed-forward manner. The last of the three angle layers f) acts as a memory due to self-excitatory connections storing the activity from the previous time step, in case no new input arrives (when the disk does not move and the event camera does not produce enough coherent spikes to bring the neurons in the Hough space above threshold). A new input, however, overwrites the stored angle by all-to-all inhibitory connectivity.

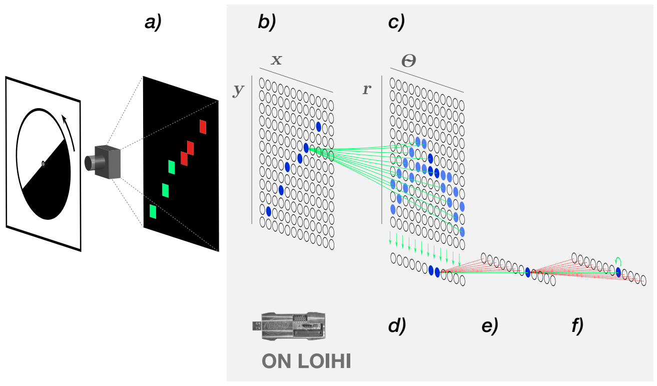


Fig. 4: Visual processing: a spiking neuronal network implementing the Hough transform on Loihi.

C. Experiments

In this work, first, we demonstrate how the neuromorphic controller is able to perform with similar speed and accuracy as the state of the art event-driven CPU controller in an extreme control scenario, such as controlling a drone moving at high speeds.

To compare the two controllers and to find remaining temporal bottlenecks in the control pipeline, we timed all stages of the control loop. The *Timing* section shows the characterization of the entire control loop for both the CPU and SNN controller.

Furthermore, we characterize the SNN controller in two ways. First, to assess the precision of the controller, a reference trajectory is imposed on the drone to bring it to different target angles. The deviation of the drone's orientation from the desired orientation was measured after 5 seconds for a duration of 10 seconds and quantified in the form of the Root Mean Square Error (RMSE). Result is compared to the controller running in software on the CPU[2].

To validate the ablity of the on-chip plasticity to adapt control parameters in an online manner, these experiments were repeated, this time with an additional weight of 125g attached to one side of the drone's arm.

The results of these experiments are shown in the section *Controller Performance: Accuracy*.

A further set of experiments compared the performance of both controllers when trying to reach a range of velocities from $-1200deg/s$ to $1200deg/s$. In order to maintain the disk at different constant speeds, a motorized axle (a drill) was attached to the axis of the disk. The rotational speed was ramped up to the desired velocity and then kept constant. The possible velocities of the disk were split into intervals of $100deg/s$. To ignore initial transient effects, the analysis of the data was done only after the first 2 seconds for a total experiment duration of 5 seconds. To quantify the performance here, the RMSE between the reading on the disk and drone encoders was calculated.

Section *Controller Performance: High Speed Controller* shows the results.

III. RESULTS

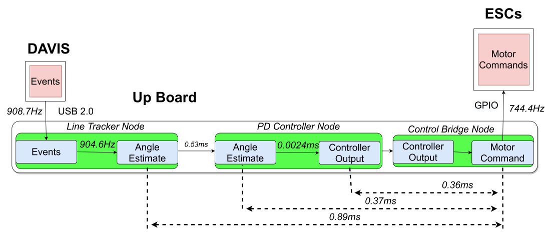
A. Timing

Figs. 5a and 5b show the computing pipeline of the control system, running on CPU and on Loihi, respectively.
The time intervals labeled on the figure show the delay induced by the respective processing stage. These latency measurements were done in a separate set of experiments than the ones done to characterize the controller behavior. Latencies are given as median over a 25 second interval.

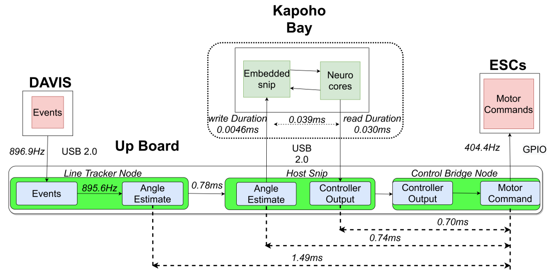
When SNN controller on-chip is used with Hough-transform computed in software, it requires 0.039ms to process an input measurement and return a control signal, as well as 0.0046ms and 0.030ms respectively for sending and reading spikes (Fig. 5b). Thus, the overall pipeline is approx. 2 times slower due to the overhead of an additional USB connection and software required to process I/O on the UP board. This can be seen in the increased time required to obtain new angle estimates (0.78ms against 0.53ms) as well the time needed for the control output to arrive at the control bridge (0.4ms against 0.1ms).

[2]CPU measurements were performed on Intel ATOM x5-Z8350 Processor with 64 bits up to 1.92GHz and 4GB of RAM (UP board).

This points to a direction for further improvement of the control pipeline by creating a direct interface between the robot's sensors and motors and the neuromorphic chip. Further, we show how the input-bottleneck can be improved by more directly connecting the DVS camera to Loihi. In future work, streaming events directly to output pins of the Kapoho Bay device and using spike-based motor control methods [37] could further reduce latency and accelerate the vision-driven control loop.



(a) Software controller Latencies



(b) Loihi controller Latencies

Fig. 5: The processing pipeline and timing of different parts of the event-driven controllers on CPU and Loihi.

B. Controller performance: Accuracy

Fig. 7 shows the performance of the software and SNN controller when holding an angular position using encoder feedback. Both controllers can solve this task: The top rows in Table **??** confirm that the obtained RMSE for the two controllers is similar, whereby a slightly increased error is noticeable for the experiments done with the SNN controller. The RMSEs were obtained by averaging over runs of 10 seconds each. This minor error is due to the lower resolution of the output commands, which results in small, high-frequency oscillations.

The plots in Fig. 7 demonstrate how the adaptation, implemented using synaptic plasticity on the neuromorphic chip, helps to remove the offset induced by an additional weight of 125g mounted on one of the UAV's arms. The plots show the roll of the drone at different target roll angles. From the blue traces in Fig. 7, as well as from the obtained errors for the software controller reported in Table **??**, it is clear that this additional weight of 125g represents an extreme disturbance. Without any means of adaptation, a persistent steady-state error of roughly 20° is observed, while much better performance is achieved with the adaptation of the SNN controller. With the added weight, the CPU controller's average error increases roughly by a factor of 2-3, while the adaptive controller manages to maintain the error on the same level as without the disturbance.

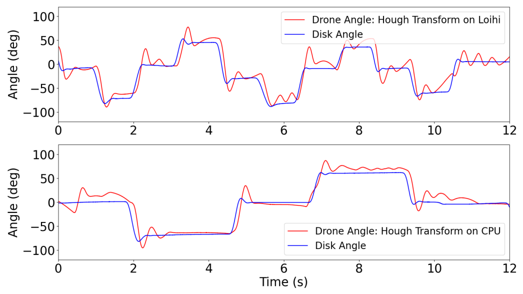


Fig. 6: **Left** Step Response using the Loihi Visual Processing as controller input. **Right** Step Response using the CPU Visual Processing as controller input

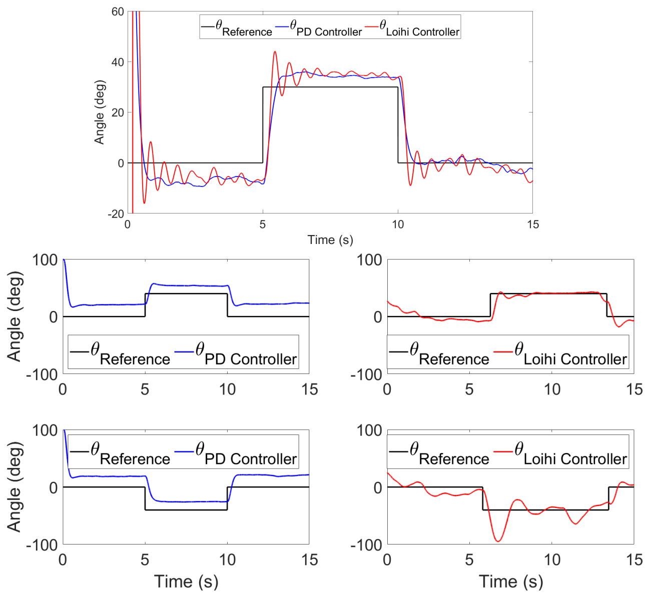


Fig. 7: **Top**: A trial run to illustrate the similar accuracy of both controllers. **Bottom**: **Left** Two trial runs using the software controller with the additional weight mounted to the drone. **Right** Same two trial runs using the adaptive SNN controller.

CPU measurements were performed on Intel ATOM x5-Z8350 Processor with 64 bits up to 1.92GHz and 4GB of RAM (UP board).

C. *Controller performance: High Speed Control*

Here, we test the ability of the drone to track the horizon rotating at different speeds, as it receives inputs from the line tracker algorithm for both the software controller and the SNN controller. As expected, the controller running on the Kapoho Bay behaves worse than the CPU controller at speeds higher than $1000 deg/s$. As described in the *Timing* section, the slower control loop, which is $2\times$ as slow, is the underlying cause for this.

These results indicate that using a neuromorphic chip as a controller for extreme scenarios is indeed a realistic possibility but that current hardware interfacing limits the attainable performance. In order to optimize the communication, next, we demonstrate how replacing one of the USB connections –from the DAVIS event camera to the Kapoho Bay– with a direct AER interface, we increase the throughput of the control loop.

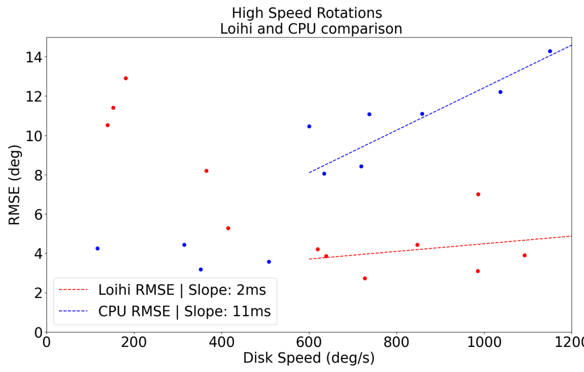


Fig. 8: RMSE in tracking the horizon for the CPU and Loihi PD controllers (both with software Hough transform) at different angular velocities.

D. *On-chip visual processing*

The Hough transform on the Loihi chip requires less than $200\mu s$ of additional processing time, which is the time for the signal to travel through the 4 layer SNN (4 time steps of $50\mu s$ duration on average) while producing an angle-estimate every time step (achieving an output rate of up to 20 kHz, compared to $<$1kHz of the event-based algorithm running on CPU). It achieves similar accuracy as the software algorithm. The SNN was currently tuned manually, and its performance can be further optimized.

Because of the favorable scaling behavior of processing speed on Loihi with the network size [20], we expect similar performance for more complex visual-processing loads.

IV. CONCLUSIONS

In this paper, we have shown that neuromorphic high-speed control is not only theoretically but also practically possible. We present the first steps into realizing the first vision-based neuromorphic controller, which may outperform a current state-of-the-art high-speed controller.

By integrating the event-based vision sensor more tightly with neuromorphic processors, we removed the main bottleneck in unleashing the full functionality of neuromorphic vision-driven control. The complexity of the visual task can be increased now, while maintaining lower latencies and lower power consumption, supported by good scaling properties of parallel and asychnronous neuromorphic hardware [20].

We also explored the possibilities which on-chip learning, enabled on the Intel's Loihi chip, may provide when dealing with external disturbances.

In the next iteration of our work, we aim to extend the direct I/O communication channels available on Intel's Kapoho Bay, which will allow us to reduce latencies and delays further, as well as to use the on-chip plastic synapses to realize non-linear adaptive control, similar to previous work on Loihi [28], but applied to high-speed agile robots.

ACKNOWLEDGMENT

We thank Adrian Paeckel and Rebhusli Team for helping with the 3D printed parts, Thomas Längle and the Robotics and Perception Group at UZH for support with the drone; INRC and NCL at Intel for support with Loihi.

REFERENCES

[1] G. Loianno, D. Scaramuzza, and V. Kumar, "Special issue on high-speed vision-based autonomous navigation of uavs," *Journal of Field Robotics*, vol. 1, no. 1, pp. 1–3, 2018.

[2] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano *et al.*, "The current state and future outlook of rescue robotics," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.

[3] D. Falanga, S. Kim, and D. Scaramuzza, "How fast is too fast? the role of perception latency in high-speed sense and avoid," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1884–1891, 2019.

[4] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, 2020.

[5] M. Mahowald, "The silicon retina," in *An Analog VLSI System for Stereoscopic Vision*. Springer, 1994, pp. 4–65.

[6] 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. [Online]. Available: http://siliconretina.ini.uzh.ch/wiki/lib/exe/fetch.php?media=lichtsteiner{_}isscc2006{_}d27{_}09.pdf

[7] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2020.

[8] R. S. Dimitrova, M. Gehrig, D. Brescianini, and D. Scaramuzza, "Towards low-latency high-bandwidth control of quadrotors using event cameras," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4294–4300.

[9] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240x180 130db 3μs latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, p. 23332341, 2014.

[10] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, P. Mostafalu, F. Brady, L. Chotard, F. LeGoff *et al.*, "5.10 a 1280$\times$ 720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86 μm pixels, 1.066 geps readout, programmable event-rate controller and compressive data-formatting pipeline," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 112–114.

[11] G. D'Angelo, E. Janotte, T. Schoepe, J. O'Keeffe, M. B. Milde, E. Chicca, and C. Bartolozzi, "Event-Based Eccentric Motion Detection Exploiting Time Difference Encoding," *Frontiers in Neuroscience*, vol. 14, no. May, pp. 1–14, 2020.

[12] S. Afshar, T. Julia Hamilton, J. Tapson, A. Van Schaik, and G. Cohen, "Investigation of event-based surfaces for high-speed detection, unsupervised feature extraction, and object recognition," *Frontiers in Neuroscience*, vol. 13, no. JAN, pp. 1–19, 2019.

[13] S.-C. Liu, B. Rueckauer, E. Ceolini, A. Huber, and T. Delbruck, "Event-driven sensing for efficient perception: Vision and audition algorithms," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 29–37, 2019.

[14] H. Akolkar, S. H. Ieng, and R. Benosman, "Real-time high speed motion prediction using fast aperture-robust event-driven visual flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[15] N. De Rita, A. Aimar, and T. Delbruck, "Cnn-based object detection on low precision hardware: Racing car case study," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 647–652.

[16] D. Gehrig, A. Loquercio, K. G. Derpanis, and D. Scaramuzza, "End-to-end learning of representations for asynchronous event-based data," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5633–5643.

[17] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "High speed and high dynamic range video with an event camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[18] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427.

[19] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, 2019.

[20] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[21] 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.

[22] 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, p. 141, 2015.

[23] M. Gehrig, S. B. Shrestha, D. Mouritzen, and D. Scaramuzza, "Event-based angular velocity regression with spiking networks," *arXiv preprint arXiv:2003.02790*, 2020.

[24] S. Glatz, J. Martel, R. Kreiser, N. Qiao, and Y. Sandamirskaya, "Adaptive motor control and learning in a spiking neural network realised on a mixed-signal neuromorphic processor," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9631–9637.

[25] R. K. Stagsted, A. Vitale, A. Renner, L. B. Larsen, A. L. Christensen, and Y. Sandamirskaya, "Event-based pid controller fully realized in neuromorphic hardware: A one dof study." in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[26] R. K. Stagsted, A. Vitale, J. Binz, A. Renner, L. B. Larsen, and Y. Sandamirskaya, "Towards neuromorphic control: A spiking neural network based pid controller for uav." in *Robotics: Science and Systems*, 2020.

[27] J. Zhao, N. Risi, M. Monforte, C. Bartolozzi, G. Indiveri, and E. Donati, "Closed-loop spiking control on a neuromorphic processor implemented on the icub," *arXiv preprint arXiv:2009.09081*, 2020.

[28] T. DeWolf, P. Jaworski, and C. Eliasmith, "Nengo and low-power ai hardware for robust, embedded neurorobotics," *arXiv preprint arXiv:2007.10227*, 2020.

[29] F. Perez-Peña, A. Morgado-Estevez, A. Linares-Barranco, A. Jimenez-Fernandez, F. Gomez-Rodriguez, G. Jimenez-Moreno, and J. Lopez-Coronado, "Neuro-inspired spike-based motion: from dynamic vision sensor to robot motor open-loop control through spike-vite," *Sensors*, vol. 13, no. 11, pp. 15 805–15 832, 2013.

[30] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. J. Douglas, and T. Delbruck, "A pencil balancing robot using a pair of aer dynamic vision sensors," in *2009 IEEE International Symposium on Circuits and Systems*. IEEE, 2009, pp. 781–784.

[31] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor," *Frontiers in neuroscience*, vol. 7, p. 223, 2013.

[32] K. Boahen, "A throughput-on-demand address-event transmitter for neuromorphic chips," in *Proceedings 20th Anniversary Conference on Advanced Research in VLSI*. IEEE, 1999, pp. 72–86.

[33] E. Mueggler, N. Baumli, F. Fontana, and D. Scaramuzza, "Towards evasive maneuvers with quadrotors using dynamic vision sensors," *Eur. Conf. Mobile Robots (ECMR)*, p. 18, 2015.

[34] Z. Jiang, Z. Bing, K. Huang, and A. Knoll, "Retina-based pipe-like object tracking implemented through spiking neural network on a snake robot." vol. 13, p. 29, May 2019.

[35] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect

lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, p. 11–15, Jan 1972.

[36] S. Seifozzakerini, W.-Y. Yau, K. Mao, and H. Nejati, "Hough transform implementation for event-based systems: Concepts and challenges." *Frontiers in Computational Neuroscience*, vol. 12, p. 103, Dec 2018.

[37] F. Perez-Pena, A. Linares-Barranco, and E. Chicca, "An approach to motor control for spike-based neuromorphic robotics," *IEEE 2014 Biomedical Circuits and Systems Conference, BioCAS 2014 - Proceedings*, pp. 528–531, 2014.