Neuromorphic Event-based Line Detection on SpiNNaker

Amélie Gruel, Adrien Vincent, Jean Martinet, Sylvain Saïghi

Neuromorphic Event-based Line Detection on SpiNNaker

Amélie Gruel*, Adrien F. Vincent*, Jean Martinet†, and Sylvain Saïghi*‡
* Laboratoire de l'Intégration du Matériau au Système, Univ. Bordeaux, Bordeaux INP, CNRS, Talence, France
† Laboratoire I3S, Université Côte d'Azur, CNRS, Sophia-Antipolis, France
‡ CNRS@CREATE, 1 Create Way, 08-10 Create Tower, Singapore 138602
Email: {amelie.gruel, adrien.vincent, sylvain.saighi}@ims-bordeaux.fr

***Abstract*—The combined use of Spiking Neural Networks (SNNs) and neuromorphic data in recent years makes for a promising solution to the challenges currently raised in computer vision. Indeed, the natural match between SNNs and event data leads to improvements in terms of biological inspiration, energy savings, latency and memory use for dynamic visual data processing, especially when such networks are implemented on neuromorphic hardware. We propose to draw advantages from these technologies to propose, to the best of our knowledge, the first end-to-end neuromorphic model for straight line detection, a standard task in robotics and computer vision. Our architecture relies on SNN intrinsic dynamics and ensures the accurate detection of moving lines recorded by an event-based camera with no learning. It reaches an overall performance of over 90 % with a limited number of neurons and synapses allowing for its deployment on the neuromorphic board SpiNNaker.**

***Index Terms*—neuromorphic, spiking neural network, event-based camera, SpiNNaker**

I. Introduction

Recent years witnessed an increasing number of studies exploring neuromorphic computing and highlighting its significant advantages in terms of processing latency, data storage and energy consumption. More and more neuromorphic sensors are being created to record and process sensory data in life-like form. A famous example of such a sensor is the event camera [1], bringing an emerging vision paradigm by mimicking the biological retina. The data recorded by this sensor make a particularly good match with spiking neural networks (SNNs), i.e. the "third generation of neural network models" [2], due to their event-driven nature.

Many recent works in embedded machine learning tend to make use of both of those scientific concepts combined [3], [4]. Indeed, "a suitable spiking neuron model with proper synaptic plasticity while exploiting event-based, data-driven updates is a major goal among neuromorphic engineers to enable computationally efficient intelligent applications" [5]. Two main approaches can be identified to this research question: either ANNs-to-SNNs conversion (ANNs: artificial neural networks) or spike-based learning, both bringing pros and cons to the table. It is however noteworthy that the first approach leads to increased latency and decreased energy efficiency whereas the spike-based approach is more bio-plausible and easier to deploy on neuromorphic hardware such as SpiNNaker [6]. Neuromorphic hardware is specifically adapted to the fast and low-power simulations of SNNs, using electronic circuits that faithfully reproduce the dynamics of neurons in real-time. It is a promising technology which still needs to overcome some challenges, such as the limitations in the number of implementable neurons and synapses, the memory consumption in data addressing and the difficulty of implementing on-board learning. For example, most works deploying SNNs on SpiNNaker perform a first "off-board" learning phase, then deploy the final architecture with the learned hyperparameters on the SpiNN board. Some researchers bypass this issue by designing architectures without any learning, specifically for application tasks where the parameters can be intuited. This methodology has been successfully applied among others to 3D reconstruction [7] and visual attention [8] performed on event data.

To the best of the authors' knowledge, there has been no prior native SNN model applied to event-based line detection even though this task is fundamental for the perception and comprehension of the visual environment [9], for example to detect the horizon or lanes on a road. Line detection is a relevant task in robot perception [10], for example in several aerial manoeuvres (landing, perching, etc). As stated in [11], "line features contain richer structure and can be more robustly extracted than punctual features such as corners".

Some approaches for event-based line detection were performed directly at the camera [12], [13] while others combined feature-extraction on RGB data with optical flow estimation on event-data [14] or made use of standard image processing algorithms such as clustering [15], iterative weighted least squares fitting [16] or an Extended Kalman Filter [11]. Interestingly, [17], [18] and [19] applied SNN models to line detection on standard RGB visual images combined with a Hough transform, while [20] introduces a SNN model combined with a Hough transform to detect contours in event data.

This work is supported by a public grant overseen by the French National Research Agency (ANR) as part of the 'Chaires IA' programme (GrAI project ANR-19-CHIA-0003) and as part of the 'PEPR IA France 2030' programme (Emergences project ANR-23-PEIA-0002). This research is part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This work is supported by the ANR project NAMED (ANR-23-CE45-0025-01).

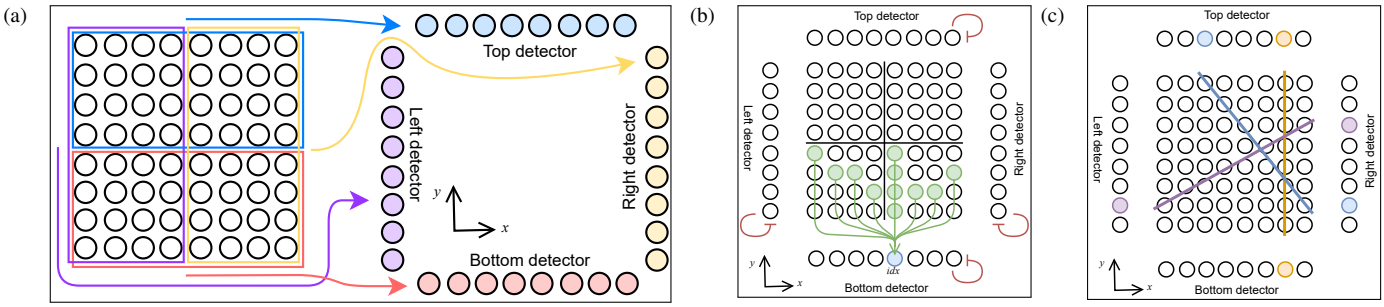

Fig. 1: Illustration of the SNN architecture implemented in this work. **a)** The global organisation of the network, with one input layer (i.e. the input sensor's pixels) and one output layer, comprising four parallel populations of neurons known as "detectors". **b)** The pattern of synaptic connections between the input and each detectors, with the specific example of connection pattern with step $k = 4$ between the output neuron within the "bottom detector" at index idx (in blue) and the input pixels belonging to the diagonals spanning from the sensor's border neuron at index idx (pixels in green). The green arrows correspond to activating synapses and the red to inhibiting synapses. **c)** Three examples of lines extrapolated from the network activation: at different timesteps, the detectors' neurons spike two by two by colour and the lines of corresponding colours are extrapolated.

We introduce in this work an SNN architecture applied to event-based straight line detection and embedded on a SpiNNaker board. We extend the "no-learning" strategy presented above to this task, by relying solely on intrinsic SNN behaviours. This model is designed with no learning phase and is light enough to allow for its easy deployment on SpiNNaker boards. This design allows for the low-latency, low-power neuromorphic detection of event lines while relying on a lower-resource architecture than a fully connected ANN approach.

II. Neuromorphic Model

A. Overall concept

The model introduced in this work detects static or moving lines in the input event data spanning the whole sensor by outputting the position of their intersections with the sensor's borders. This mechanism is implemented using one output layer comprising four parallel populations of neurons, each corresponding to one border of the sensor and named accordingly. As pictured in Fig. 1a, each of those populations or "detector" corresponds to a different half of the sensor: the "top detector" supervises the top half (in blue in the figure), the "right" one the right half (in yellow), the "bottom" one the bottom half (in red) and the "left" one the left half (in purple). Each detector supervises its corresponding half and detects any line within this half-sensor and crossing the corresponding border, thanks to a specific pattern of connections (detailed in section II-B and Fig. 1b). When a line crossing the border at the position $(x_{\text{intersection}}, y_{\text{intersection}})$ is detected within a half-sensor, the neuron at the corresponding index idx in the corresponding detector spikes as depicted in the example presented in Fig. 1c. Depending on the detector, idx equals either $x_{\text{intersection}}$ for "top" and "bottom" detectors, or $y_{\text{intersection}}$ for "left" or "right". For each line in the input event data, two different detectors activate thanks to this mechanism; from this activity, one can then extrapolate the line extrema, thus its position.

B. Architecture

As stated above, each detector supervises its half-sensor thanks to a specific activation pattern: the Leaky-Integrate-and-Fire neurons [**?**] at index idx in a detector are activated by the input neurons (assimilated to the pixels of a retinomorphic sensor) belonging to diagonal patterns expanding from the neurons situated at index idx on the corresponding input sensor's border (see Fig. 1b, with the example of the blue neuron at index idx in the Bottom detector). This is repeated for each neuron of each detector, with the same interval between diagonals in the pattern. To increase the selectivity of each detector, a Winner-Takes-All mechanism is applied to each of their neurons: the activation of any neuron will laterally inhibit the other neurons in the population.

It should be noted that the diagonals are computed as lines spanning from each neuron of one sensor's border to either of the three other borders of the half-sensor. In other words, a pattern of connections spanning from a neuron on the bottom border would connect it with each neuron on the borders of the bottom half of the left and right borders as well as with the neurons on the horizontal line cutting the sensor in half (in black in Fig. 1b). The proposed model allows to implement a pattern of connections which selects a diagonal for every k potential diagonals, with $k = 1$ for a pattern encompassing all potential diagonals ($k = 4$ in Fig. 1b). The higher the value k is set to, the lower the number of synapses to implement and the lighter the architecture becomes.

A first iteration of this work used an equal activation weight ω for all input neurons. However, it was immediately made clear that such connectivity quickly led to an unwanted lateral activation, the saturation of detectors not concerned by the input and the false detection of lines where there are none. Indeed, if all connections activate strongly the detectors, then in the example depicted in Fig. 1b, a hypothetical horizontal line in the upper part of the bottom half of the sensor would saturate the bottom detector's neurons. To overcome this

issue, the weights of each diagonal connection are inversely proportional to the Euclidian distance (in pixels) between the input sensor's neurons and the detectors' neurons: the further away the input neuron is from the sensor's border, the lower the weight of the corresponding synaptic connection. The weights ω are calculated under Eq. 1 so that their sum ω_{sum}, i.e. the total activation received by the input neuron when a line is recorded at the pixels belonging to the corresponding diagonal, is known and can be adjusted:

$$\begin{aligned} \omega &= \max(\omega_{\text{smallest}} \times (l_{\text{diago}} - d), \omega_{\text{min}}) \\ \text{with } \omega_{\text{smallest}} &= 2 \times \omega_{\text{sum}} / (l_{\text{diago}} \times (l_{\text{diago}} + 1)) \end{aligned} \quad (1)$$

where ω_{smallest} is the smallest weight a connection can have, calculated using the formula for the sum of increasing integers; l_{diago} is the length of the diagonal of interest; d the distance between the input neuron of interest and the point of intersection between the diagonal and the border of the sensor; ω_{min} the lower bound applied to the activation weight.

C. Implementation

The main objective of this work is the implementation of event-based line detection on the neuromorphic hardware SpiNNaker [6], as a first step for an embedded, low-latency and low-energy application. This objective dictated both our choices of implementing such a mechanism with no learning as well as using such a light architecture. Indeed, any SpiNNaker platform is limited in terms of synapses (at most 1000 synapses per output neurons) and neurons, depending on the board (SpiNN-3 can simulate at most 18,000 neurons, and SpiNN-5 at most 195,000). As SpiNNaker does not require input neurons to be simulated, the deployment of our architecture on this board requires only $4S$ neurons (with $S \times S$ the sensor size) and, in the heavier case scenario (i.e. $k = 1$), $\frac{S^2}{2}$ incoming synapses per output neuron. Comparing this to a theoretical fully connected approach, which would require $6S^2 - 19S + 20$ neurons and S^2 synapses per output neuron, this SNN architecture allows for an efficient embedded application and could more easily be applied to high-resolution event cameras, such as the Prophesee One Megapixel Camera [21], [22], than a fully connected "dense" approach (see Fig. 2).

The results below were obtained on CPU using the NEST simulator [23] interfaced with PyNN [24], then deployed on the publicly available online SpiNNaker platform [25].

III. Experimental results

A. Input data

In this section, the model introduced in section II is validated by simulation means on a synthetic event dataset. To the best of the authors' knowledge, there are no real-life event-based datasets with line recording and labelling. A custom-made dataset was thus designed where events are generated belonging to input diagonal lines, randomly spanning the sensor. Each line shifts by one pixel for each timestep, over $t_{pattern}$ timesteps. To ease the calculation of the model's performance, each line apparition is followed by an empty screen for $t_{pattern}$ timesteps. The results presented below are obtained from a dataset of 100 diagonal lines, spanning a synthetic sensor of size 28×28 pixels with $t_{pattern} = 10$ timesteps. The sensor size is similar to the active area recorded in N-MNIST, a benchmark neuromorphic dataset [26].

Parameters	Values
Width & height	28
$t_{pattern}$	10 timesteps
Membrane potential v_{rest} and v_{reset}	–60 mV
Neuronal threshold v_{thresh}	–30 mV
Membrane time constant τ_m	0.1 ms
Refractory period τ_r	1 ms
Activation weight (diagonal connections) ω	4
Lower bound of the activation weight ω_{min}	0.01
Inhibition weight (WTA) ω_{WTA}	1
Step k between diagonal connections	9

TABLE I: Neuronal and synaptic parameters of our model

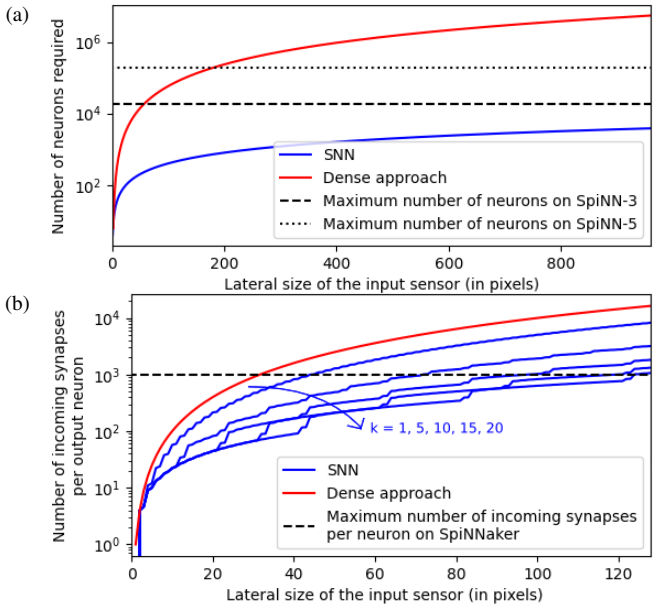

Fig. 2: Comparison between the number of neurons (**a**) and number of incoming synapses per neuron (**b**) required by a fully connected model (in red) and our model with a varying step k between neurons (in blue). The upper bounds on SpiNN-3 and SpiNN-5 boards are illustrated in black.

B. Performance of moving line detection

Fig. 3a presents the results obtained for different hyperparameters on the input dataset described above. On the x- and y-axis respectively, the parameters "step k between diagonal connections" and "weight ω_{sum}" correspond to the parameter k and ω_{sum} introduced in section II-B.

The performance is measured using three standard metrics [27]. The precision evaluates whether the actual spike activity matches with what it should have been, and the recall evaluates whether all the spikes that should have happened did happen. The F1-score measures the overall performance of the network, therefore evaluated by comparing the expected activity (i.e. the input spikes) to the obtained activity (i.e. the output spikes) for each line detector.

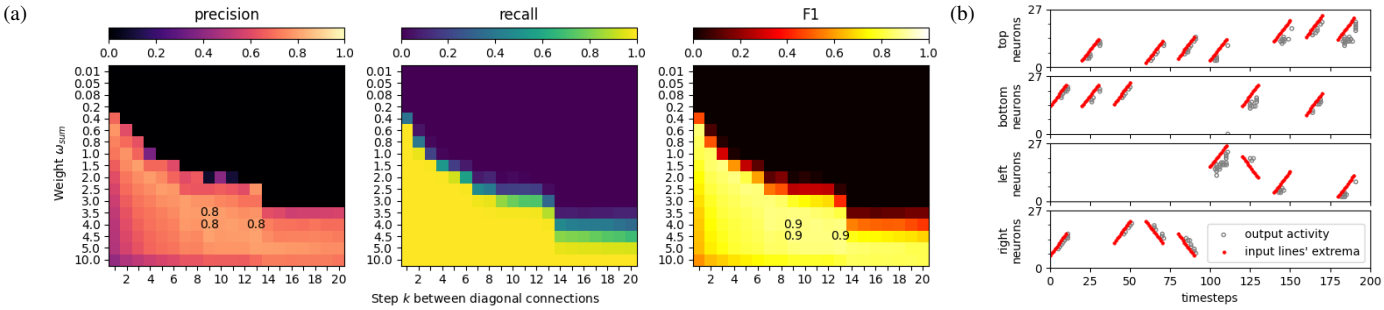

Fig. 3: Network performance of line detection. **a)** Performance according to the weight ω_{sum} and the step k between diagonal connections. **b)** Input lines' extrema (in red) and output activity of the network (in grey) to allow for qualitative assessment.

These results clearly indicate an inversely proportional relation between the number of diagonal connections (i.e. the inverse of k) and the total activation weight required. Indeed, the bigger the gap between the diagonal connections, the bigger the total weight necessary to apply the input activity to the output detector layers. Additionally, the "recall" plot shows a steep boundary between the hyperparameters sets which do or do not allow for the correct activity in the output detectors: when the recall is close to 0, the output layers did not activate since the total activation weight is not high enough compared to the number of diagonal connections. However, once the minimum weight is reached, the recall is always perfect. Depending on the parameters, these correct activations can quickly be accompanied by a significant number of incorrect activations: this is either due to the activation of neurons at an incorrect index or to the activation of detectors not corresponding to the intersection of the input line with the sensor's borders. This last case is a residue of the lateral saturation phenomenon described in section II-B and mostly overcome using varying weights in the diagonal connections; it can take place when the weight is too high compared to the number of diagonal connections and affect the precision of the model, for example at the bottom left of the "precision" plot in Fig. 3a. Thanks to the F1-scores, one can easily pick the most appropriate parameters for the ideal neuromorphic line detection using this model.

Fig. 3b presents the results obtained on a sample of 10 moving lines, over a sensor of size 28×28 pixels. Each red dot corresponds to the extrema of the input lines, i.e. their intersection with the sensor's border, moving over time over 10 timesteps. Grey dots illustrate the detectors' spikes output by the model deployed on SpiNNaker using the parameters defined in Tab. I, according to the results presented in Fig. 3a, and allows to visually assess the quality of line detection.

The deployment of our model on SpiNNaker consumes only 187.3 mJ per timestep of processed input data according to the platform's reports.

C. Resistance to noise and event-drop

As presented in the previous section, our model achieves good performance in detecting lines in event data. However, to assess its potential application with real-life data, we compare its performance on input event data subject to noise and event-drop, using functions implemented within the Tonic framework [28]. Fig. 4 illustrates the evolution of the model performance according to an increasing noise (figure on the left) and increasing probability of dropping events (figure on the right). Our architecture is tolerant to noise up to a certain point: up until a probability of noise of 1 % per timestamp per pixel, it maintains a good performance; once this threshold is crossed, the precision drops significantly (probably due to an increased lateral saturation behaviour). Concerning the event drop, our model conserves a similar performance up until a probability of dropping events of 10 % (i.e. an event does not take place when it should in the input sensor one time out of 10); once this threshold is crossed, the precision quickly decreases due to the limited amount of input activity.

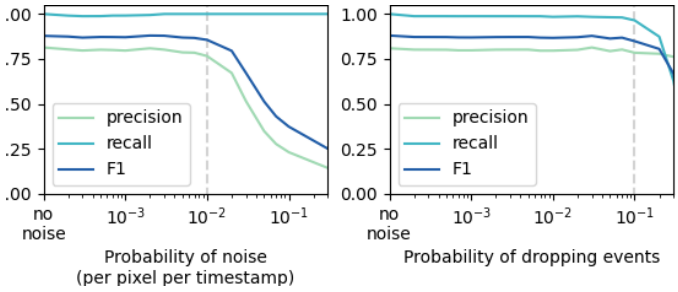

Fig. 4: Evolution of performance according to the probability of added noise or dropped events.

IV. Conclusion

To the best of the authors' knowledge, this work introduces the first end-to-end neuromorphic model for line detection, applied to event data. We ensure the accurate detection of line reaching an overall performance of over 90 % with a limited number of neurons and synapses. Our model shows a good robustness to noise (up to 1 %) and event drop (up to 10 %). Its deployment on SpiNNaker leads to a power consumption of 187.3 mJ per timestep of incoming data.

In future works, the model could be made more flexible by applying a weight adaptation rule on ω similarly to [8]; to be noted that such a strategy has the downside of higher latency and lesser ease to deploy on SpiNNaker. Another avenue of research would be to validate our model on real-life event data, for example using the DET dataset [29].

REFERENCES

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15 us latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, 2008.

[2] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[3] G. Gallego, T. Delbruck, G. Orchard, and al., "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[4] D. V. Christensen, R. Dittmann, B. Linares-Barranco, *et al.*, "2022 roadmap on neuromorphic computing and engineering," *Neuromorphic Computing and Engineering*, vol. 2, p. 022501, may 2022.

[5] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[6] S. B. Furber *et al.*, "Overview of the SpiNNaker system architecture," *IEEE Transactions on Computers*, 2013.

[7] G. Dikov, M. Firouzi, F. Röhrbein, J. Conradt, and C. Richter, "Spiking cooperative stereo-matching at 2 ms latency with neuromorphic hardware," pp. 119–137, 07 2017.

[8] A. Gruel, A. Vitale, J. Martinet, and M. Magno, "Neuromorphic event-based spatio-temporal attention using adaptive mechanisms," in *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2022.

[9] C. Blakemore and G. COOPER, "Development of the brain depends on the visual environment," *Nature*, vol. 228, pp. 477–8, 10 1970.

[10] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A stereo SLAM system through the combination of points and line segments," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 734–746, 2019.

[11] A. Gómez Eguíluz, J. Rodríguez-Gómez, J. Martínez-de Dios, and A. Ollero, "Asynchronous event-based line tracking for time-to-contact maneuvers in UAS," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5978–5985, 2020.

[12] C. Posch, M. Hofstatter, D. Matolin, G. Vanstraelen, P. Schon, N. Donath, and M. Litzenberger, "A dual-line optical transient sensor with on-chip precision time-stamp generation," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pp. 500–618, 2007.

[13] C. Brändli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbruck, "ELiSeD — an event-based line segment detector," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pp. 1–7, 2016.

[14] K. Li, D. Shi, Y. Zhang, R. Li, W. Qin, and R. Li, "Feature tracking based on line segments with the dynamic and active-pixel vision sensor (DAVIS)," *IEEE Access*, vol. 7, pp. 110874–110883, 2019.

[15] L. Everding and J. Conradt, "Low-latency line tracking using event-based dynamic vision sensors," *Frontiers in Neurorobotics*, vol. 12, 02 2018.

[16] D. Reverter Valeiras, X. Clady, S.-H. Ieng, and R. Benosman, "Event-based line fitting and segment detection using a neuromorphic visual sensor," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 4, pp. 1218–1230, 2019.

[17] Q. Wu, T. Mcginnity, L. Maguire, A. Belatreche, and B. Glackin, "Edge detection based on spiking neural network model.," pp. 26–34, 01 2007.

[18] Q. Wu, T. M. McGinnity, L. Maguire, G. Valderrama-Gonzalez, and J. Cai, "Detection of straight lines using a spiking neural network model," in *2009 Fifth International Conference on Natural Computation*, vol. 2, pp. 385–389, 2009.

[19] X. Li, Q. Wu, Y. Kou, L. Hou, and H. Yang, "Lane detection based on spiking neural network and hough transform," in *2015 8th International Congress on Image and Signal Processing (CISP)*, pp. 626–630, 2015.

[20] S. Aspragkathos, E. Ntouros, G. Karras, B. Linares-Barranco, T. Serrano-Gotarredona, and K. Kyriakopoulos, "An event-based tracking control framework for multirotor aerial vehicles using a dynamic vision sensor and neuromorphic hardware," *IROS*, 2023.

[21] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, P. Mostafalu, F. Brady, L. Chotard, F. LeGoff, H. Takahashi, H. Wakabayashi, Y. Oike, and C. Posch, "5.10 a 1280×720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86μm pixels, 1.066GEPS readout, programmable event-rate controller and compressive data-formatting pipeline," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 112–114, 2020.

[22] E. Perot, P. de Tournemire, D. Nitti, J. Masci, and A. Sironi, "Learning to detect objects with a 1 megapixel event camera," *CoRR*, vol. abs/2009.13436, 2020.

[23] M.-O. Gewaltig and M. Diesmann, "NEST (NEural Simulation Tool)," 2007.

[24] A. P. Davison, D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "PyNN: A common interface for neuronal network simulators.," *Fr. in Neuroinformatics*, vol. 2, 2009.

[25] A. P. Davison, E. Müller, S. Schmitt, B. Vogginger, D. Lester, and T. Pfeil, "The spinnaker "many core" system — hbp neuromorphic computing platform guidebook (wip)," Aug 2022.

[26] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, p. 437, 2015.

[27] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *CoRR*, vol. abs/2010.16061, 2020.

[28] G. Lenz, K. Chaney, S. B. Shrestha, O. Oubari, S. Picaud, and G. Zarrella, "Tonic: event-based datasets and transformations.," jul 2021. Documentation available under https://tonic.readthedocs.io.

[29] W. Cheng, H. Luo, W. Yang, L. Yu, S. Chen, and W. Li, "DET: A high-resolution dvs dataset for lane extraction," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1666–1675, 2019.