Neuromorphic computation using event-based sensors : from algorithms to hardware implementations

Germain Haessig

THÈSE DE DOCTORAT

EN VUE DE L'OBTENTION DU TITRE DE DOCTEUR EN ROBOTIQUE DE SORBONNE UNIVERSITÉS

ECOLE DOCTORALE 391 : SCIENCES MÉCANIQUES, ACOUSTIQUE, ELECTRONIQUE ET ROBOTIQUE DE PARIS

NEUROMORPHIC COMPUTATION USING EVENT-BASED SENSORS : FROM ALGORITHMS TO HARDWARE IMPLEMENTATIONS

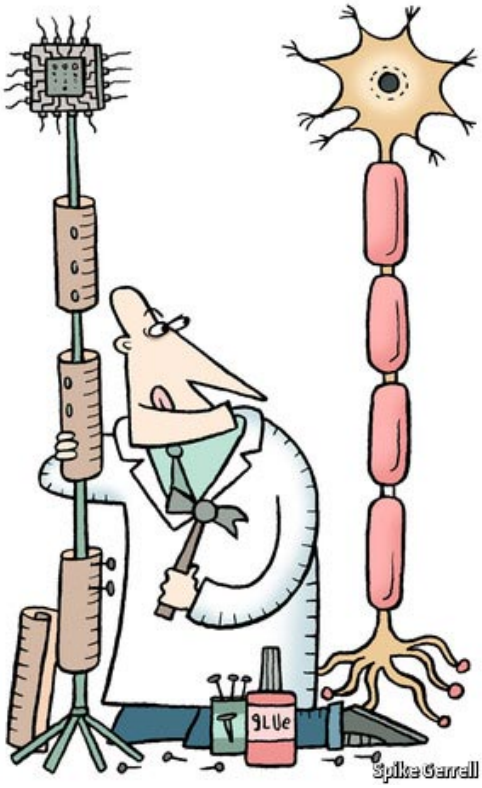
PRÉSENTÉE PAR
GERMAIN HAESSIG

SOUS LA DIRECTION DE
PR. RYAD B. BENOSMAN

PRÉSENTÉE ET SOUTENUE PUBLIQUEMENT LE 14 SEPTEMBRE 2018

DEVANT UN JURY COMPOSÉ DE :

Associate Pr.	Sio Hoi Ieng	*Examinateur*
Pr.	Stéphane Régnier	*Examinateur*
Pr.	Bernabé Linares-Barranco	*Rapporteur*
Pr.	Giacomo Indiveri	*Rapporteur*
Pr.	Ryad B. Benosman	*Directeur de thèse*



Germain Haessig : *Neuromorphic computation using event-based sensors : from algorithms to hardware implementations*, Ph.D. Thesis, September 2018.

Contact : germain.haessig@inserm.fr

Institut de la Vision
Equipe Vision et Calcul Naturel
17, Rue Moreau
75012 Paris, France

ABSTRACT

This thesis is about the implementation of neuromorphic algorithms, using, as a first step, data from a silicon retina, mimicking the human eye's behavior, and then evolve towards all kind of event-based signals. These event-based signals are coming from a paradigm shift in the data representation, thus allowing a high dynamic range, a precise temporal resolution and a sensor-level data compression. Especially, we will study the development of a high frequency monocular depth map generator, a real-time spike sorting algorithm for intelligent brain-machine interfaces, and an unsupervised learning algorithm for pattern recognition. Some of these algorithms (Optical flow detection, depth map construction from stereovision) will be in the meantime developed on available neuromorphic platforms (SpiNNaker, TrueNorth), thus allowing a fully neuromorphic pipeline, from sensing to computing, with a low power budget.

RÉSUMÉ

Cette thèse porte sur l'implémentation d'algorithmes événementiels, en utilisant, dans un premier temps, des données provenant d'une rétine artificielle, mimant le fonctionnement de la rétine humaine, pour ensuite évoluer vers tous types de signaux événementiels. Ces signaux événementiels sont issus d'un changement de paradigme dans la représentation du signal, offrant une grande plage dynamique de fonctionnement, une résolution temporelle importante ainsi qu'une compression native du signal. Sera notamment étudiée la réalisation d'un dispositif de création de cartes de profondeur monoculaires à haute fréquence, un algorithme de tri cellulaire en temps réel, ainsi que l'apprentissage non supervisé pour de la reconnaissance de formes. Certains de ces algorithmes (détection de flot optique, construction de cartes de profondeur en stéréovision) seront développés en parallèle sur des plateformes de simulation neuromorphiques existantes (SpiNNaker, TrueNorth), afin de proposer une chaîne de traitement de l'information entièrement neuromorphique, du capteur au calcul, à faible coût énergétique.

AVANT-PROPOS

Au seuil de ce parcours scientifique, je souhaiterais, avec beaucoup d'humilité et tout en ayant conscience de mon peu de recul, évoquer plusieurs aspects de mon travail d'une manière plus personnelle. Il s'agira donc ici plus de mon expérience de la thèse, de ces trois années de recherche, souvent passées à courir après le temps, parfois à m'enthousiasmer devant un résultat bien meilleur qu'espéré, mais aussi à être exaspéré face à l'infructuosité de tel algorithme ou de l'échec de tel projet. Ainsi va la science, m'avait on dit.

La thèse est un peu comme un chemin sinueux, où l'on se perd facilement, où les bifurcations, si elles existent, font office de sentes cachées derrière un fourré. Se repérer devient alors un art, que l'on apprend à maîtriser au fil des années, jusqu'au moment de transmettre le peu que l'on a appris.

Enseigner durant ces années a été à la fois une échappatoire, mais aussi une source de satisfaction. Cette dure science de la transmission des connaissances, d'éveiller la curiosité de l'auditoire, continuera d'enrichir ma vision d'un domaine que l'on pense, à tort, cerner. Quoi de plus beau qu'un regard, peut-être naïf, mais aussi différent, sur un sujet particulier ?

Avant tout, cette thèse aura été l'occasion de découvrir un domaine qui, s'il n'a pas encore acquis ses lettres de noblesse, se révèle être un champ dynamique, en plein essor, se basant sur des technologies arrivant à maturité, et véhiculant avec lui un ensemble de belles promesses, dont je ne peux que souhaiter l'accomplissement. J'aspire à continuer à travailler dans ce milieu, ainsi qu'à faire bon usage de l'introspection et de la patience que cette aventure m'a apporté.

~ ~

Il va de soi que cette aventure n'aurait été possible sans le soutien indéfectible de nombreuses personnes, que je vais essayer de remercier ici, en tentant d'en omettre le moins possible.

Tout d'abord, à mes parents, sans qui cette aventure n'aurait jamais eu lieu. Pour ce goût de la curiosité que vous m'avez inculqué, cet état d'esprit ne pouvant prendre pour argent comptant ce qui est parfois communément accepté.

Ensuite, à Elise, ma sœur. Tu as été présente tout au long de l'aventure, avec tes mots de soutien, tes sourires et ces heures, malheureusement bien trop peu nombreuses, passées à refaire le monde.

À Julien et Loïc, soutiens inébranlables depuis de bien longues années maintenant. Nous avons vécu de beaux moments, et de belles aventures nous attendent encore, je l'espère. Et bien entendu à Laura et Lucie, qui ont su me supporter et accepter que je leur emprunte leur compagnon pour de longues sorties en montagne. Les années passent, mais vous revoir me fait à chaque fois un bien fou.

À tous mes collègues, passés ou présents, qui ont su faire de cette dure quête un doux moment. Camille, Quentin, Francesco et Xavier L., mes premières racines dans l'équipe, qui se sont, au fil des années, envolées vers de nouvelles aventures. Je garderai toujours en mémoire nos discussions interminables autour de la machine à café. Merci pour votre accueil chaleureux au sein de cette équipe.
Puis, à la nouvelle génération, avec laquelle j'ai traversé les eaux troubles de la recherche, en espérant trouver une issue à nos heures de réflexions.
Bien entendu, à la *Team Arkose*, pour ces heures à respirer de la magnésie, s'esquinter les doigts sur des prises bien souvent trop petites, rigoler et profiter de belles journées pour aller tâter du rocher. Pour votre soutien lorsque mon corps n'a voulu suivre, et vos encouragements lorsqu'il m'a été possible de reprendre. En espérant vous retrouver aux détours d'un rocher très rapidement.
À Paul, Kevin A. et Xavier B., qui sont devenus avec le temps bien plus que des collègues.
Aux différents bars que nous avons fréquentés avec assiduité, à râler, refaire le monde ou tout simplement discuter.
À la nouvelle génération, qui saura faire face à l'adversité et porter ce magnifique domaine encore plus haut.

À Piotr Dudek, qui a pris du temps pour me faire découvrir le doux monde du semi-conducteur, du design au layout.

À Garrick Orchard, pour ce projet un peu fou de sortir un papier après Telluride.

À Laurent Cabaret, pour son soutien lors de mon monitorat, sa confiance et la liberté qu'il m'a laissée. Enseigner à tes cotés a été un réel plaisir.

À Jean-Pierre Barbot et Dominique Placko, deux de mes enseignants à l'École Normale, qui ont su éveiller ma curiosité sur l'électronique analogique, et pour ces nombreux projets encadrés avec vous.

À Stéphane Laveau, et toute l'équipe de Prophesee, avec qui j'ai apprécié travailler, et passé des heures à bricoler des démonstrateurs de cette fabuleuse technologie.

À toute la communauté Neuromorph', pour les longues heures à discuter lors de nos diverses rencontres, à CapoCaccia ou Telluride.

Et bien sûr à Ryad, mon directeur de thèse, qui n'a cessé de croire en moi et m'a poussé à toujours chercher plus loin.

Bien à vous.

Simplicity is the ultimate sophistication.

LEONARDO DA VINCI

CONTENTS

PUBLICATIONS

JOURNALS

Haessig, G., Cassidy, A., Alvarez, R., Benosman, R., & Orchard, G. (2018). Spiking Optical Flow for Event-based Sensors Using IBM's TrueNorth Neurosynaptic System. IEEE Transactions on Biomedical Circuits and Systems, (99), 1-11.

Haessig, G., Berthelon, X., Ieng, S.H., and Benosman, R (2018). A Spiking Neural Network Model of Depth from Defocus for Event-based Neuromorphic Vision. *Submitted*.

Haessig, G., Gehere, K., and Benosman, R (2018). Spikes decoding spikes : an event-based framework for real-time unsupervised spike sorting. *Submitted*.

Haessig, G., Galluppi, F., Lagorce, X., and Benosman, R (2018). Neuromorphic networks on the SpiNNaker platform. *In preparation*.

CONFERENCES

Haessig, G., & Benosman, R. (2018, May). A sparse coding multi-scale precise-timing machine learning algorithm for neuromorphic event-based sensors. In : *Micro-and Nanotechnology Sensors, Systems, and Applications X (Vol. 10639, p. 106391U). International Society for Optics and Photonics.*

1 INTRODUCTION

Figure 1 – Sea turtle. Flickr photo by Philippe Guillaume.

This is a sea turtle. A (very) old animal, that is, every year, migrating from Africa to America, a 6000km journey that takes approximately 2 months. For years, scientists have been looking for the reason of this travel. They found that the beaches in both destinations are identical. Same climate. Same amount of food. And even same food. So why should this turtle take part in such a journey, costing time and energy, to reach a point that is similar to the starting one ? Biologists argued that the turtle is following the gulf stream to grab some plankton, explaining why the chosen path is not a straight one. But plankton can be found in closest places. For now, no plausible clues have been found.

But...

The sea turtle is a 110-million-years-old animal. Fossils have been found in both sides of the Atlantic Ocean, showing that this journey has been part of the turtle life for a long time. What was the shape of the world 110 million years ago ? Africa and America were so close that they formed an unique continent. At that time, the inferior *Cretaceous* (146million-100million years ago), the two identical beaches were probably a few miles apart. One might have sunrise, the other one sunset. Then, the continent drift slowly took these two places apart. Meanwhile, the turtle kept doing its routine, increasing slowly the distance required to reach the opposite coast, which is written in its genes. This may explain why, 110 million years later, this turtle is traveling across an ocean to reach its destination. [1]

I do not claim this theory is the good one. Better ones might exist but this story shows us one important thing : sometimes, it is impossible to understand the reasons of a given behavior just by observing it. We have to go back to the origins of the phenomenon, put it in perspective, understand

1. Personal conversation with Florian Engert, Professor of Molecular and Cellular Biology, Harvard.

the growth and development of living systems in order to fully explain a phenomenon. This has to be remembered.

MOTIVATION

My first electrical engineering lecture was more than a decade ago. And, as in most of the following ones, it started with Moore's law. That the number of transistor in a given device will, and should, increase by a factor of 2 every 2 years[85]. This has been true for a long time (factor 1.96 between 1971 and 2001), but nowadays, an inflection in this curve can be seen. Why ? This thesis will not be about electrical engineering, but during these three years, I found some hints that became, year after year, some kind of a motivation leading my work.

Since 2004, the frequency of Intel processors has leveled around 3 GHz[135]. Nowadays, performance is given by the number of cores, and no more the clock frequency. Lot of cores means lot of transistors. Even if the energy required by a single transistor to switch is small, integration of billions of them in a single die made the power consumption a major consideration in design. Transistors also became smaller. This size reductions had the effect that the transistors are not exactly totally off, a (very) small current continuously leaking through it. Again, this is not a concern for a single unit. But with dies containing billion transistors, the equation changed. State of the art GPUs integrate 21 billion transistors (NVIDIA GV100 Volta, 12nm, 2017 [2]) or FPGAs with 50 billion (Xilinx Everest, 7nm, 2018 [3]). But that scaling, according to the self-fulfilling Moore's prophecy, couldn't last forever : transistors can not be smaller than atoms. This is physics, and it can not be bypassed.

So, thinking about new approaches is needed.
The neuromorphic community, established by Carver Mead and Misha Mahowald in the late 80's[81], has been trying to find answers to these issues. They developed the first prototype of a silicon retina, mimicking the behavior of the human eye[70]. Misha Mahowald received, in 1992, Caltech's Milton and Francis Clauser Doctoral Prize for her thesis' originality and "potential for opening up new avenues of human thought and endeavor". Since, the neuromorphic community is trying to find new approaches, some *paradigm shift*, to renew our way of seeing *Perception* and *Computation*.

But what does Neuromorphic mean ?

> Neuromorphic engineering, also known as neuromorphic computing, is a concept developed by Carver Mead in the late 1980s, describing the use of very-large-scale integration (VLSI) systems containing electronic analog circuits to mimic neuro-biological architectures present in the nervous system. In recent times the term neuromorphic has been used to describe analog, digital,

2. https://www.nvidia.com
3. https://www.xilinx.com

> mixed-mode analog/digital VLSI, and software systems that implement models of neural systems (for perception, motor control, or multisensory integration). ”
>
> (*Wikipedia* [4])

The current trend is on Convolutionnal Neural Networks. State of the art dedicated implementations can tackle the problem of computation with 1pJ per addition and 5pJ per multiplication[34]. But these networks need Random Access Memory (RAM) to all the values (membrane potential, weights). SRAM (Static RAM) access costs 10pJ, DRAM (dynamic RAM) 1nJ. Most of the energy is then spent on memory access, not on computation. Memory is then the bottleneck. I do not claim that CNNs can not solve problems, because they do [61][32], but they are not the most efficient way of doing this. The brain is 30W. Lee Sedol is burning 30W for his brain. DeepMind, with its 1202 CPUs and 176 GPUs, can be estimated at 275kW[116], just for learning... With its 30W, Lee Sedol can also read, speak, interact with other people, and many other things that DeepMind is not able to do. This is why *we* are here, and why this field has a beautiful story to write.

DISSERTATION STRUCTURE

This dissertation is organized in two parts. First (Chapters 2,3 and 4), I will introduce new algorithms handling event-based signals. Chapter 2 presents an event-based approach to monocular depth estimation, Depth From Focus, using an event-based sensor and a liquid lens, allowing low power and high frequency depth map construction. Chapter 3 uses some sparse coding tools already existing in the literature and applies them to event-based machine learning algorithm, in order to drastically reduce the number of required prototypes, and thus allowing implementation in hardware with limited resources. Chapter 4 applies a modified version of the machine learning algorithm presented in the second chapter to mono-dimensional signals, for online unsupervised spike sorting tasks. The second part (Chapters 5 and 6) details some implementations of neuromorphic networks on two dedicated platforms : Chapter 5 details the implementation of a spiking neural network to compute optical flow on TrueNorth, IBM's neurosynaptic platform. Chapter 6 is about the implementation of the same optical flow network, and a stereovision network, on the SpiNNaker platform.

4. https://en.wikipedia.org/wiki/Neuromorphic_engineering

2 DEPTH FROM DEFOCUS

Depth from defocus is an important mechanism that enables vision systems to perceive depth. While machine vision has developed several algorithms to estimate depth from the amount of defocus present at the focal plane, existing techniques are slow, energy demanding and mainly relying on numerous acquisitions and massive amounts of filtering operations on the pixels' absolute luminance value. Recent advances in neuromorphic engineering allow an alternative to this problem, with the use of event-based silicon retinas and neural processing devices inspired by the organizing principles of the brain. In this paper, we present a low power, compact and computationally inexpensive setup to estimate depth in a 3D scene in real time at high rates that can be directly implemented with massively parallel, compact, low-latency and low-power neuromorphic engineering devices. Exploiting the high temporal resolution of the event-based silicon retina, we are able to extract depth at 100Hz for a power budget lower than a 200mW (10mW for the camera, 90mW for the liquid lens and ~ 100mW for the computation). We validate the model with experimental results, highlighting features that are consistent with both computational neuroscience and recent findings in the retina physiology. We demonstrate its efficiency with a prototype of a neuromorphic hardware system and provide testable predictions on the role of spike-based representations and temporal dynamics in biological depth from defocus experiments reported in the literature.

2.1 INTRODUCTION

The complexity of eyes' inner structure implies that any visual stimuli from natural scenes contains a wide range of visual information, including defocus. Several studies have shown that defocus is essential in completing some tasks and more specifically for depth estimation [48, 130]. Although a large body of research on Depth From Defocus (DFD) exists since the early 60's, there is currently a gap between the information output from biological retinas and the existing literature both in the vision science and computer vision that uses images as the sole source of their studies. Although images are perfect to display static information, their use in acquiring dynamic contents of scenes is far from being optimal. The use of images implies a stroboscopic acquisition of visual information (unknown to biological systems) at a low sampling frequency. They are thus unable to describe the full dynamics of observed scenes. On the other hand, retinal outputs are massively parallel and data-driven: ganglion cells of biological retinas fire asynchronously according to the information measured in the scene [42, 13] at millisecond precision. Recent neuroscience findings show that this temporal precision can also be found in other subcortical areas, like the lateral geniculate nu-

cleus (LGN) [65, 108] and the visual cortex [71]. The last decade has seen a paradigm shift in neural coding. It is now widely accepted that precise timing of spikes open new profound implications on the nature of neural computation [110, 66]. The information encoded in the precise timing of spikes allows neurons to perform computation with a single spike per neuron [125]. Initially supported by theoretical studies [124], this hypothesis has been later confirmed by experimental investigations [55, 106].

Here, we present a novel approach to the depth from defocus, inspired by biological retina ouput, which is compatible with ultra low latency and low power neuromorphic hardware technologies [21]. In particular, we exploit advances made in both mixed signal Analog/Digital VLSI technology and computational neuroscience which enabled us to combine a new class of retina-like artificial vision sensors with brain-inspired spiking neural processing devices to build sophisticated real-time event-based visual processing systems [88, 53, 113]. We show how precise timing of spiking retinas allows the introduction of a novel, fast and reliable biologically plausible solution to the problem of estimating depth from defocus directly from the high temporal properties of spikes.

Silicon retinas located at the core of the hereby presented system are a novel piece of hardware which do not sense scenes as a serie of frames. Conventional cameras wastefully record entire images at fixed frame rates(30-60Hz) that are too slow to match the temporal sub-millisecond resolution of human senses. Silicon retinas are asynchronous and clock-less, every pixel is independent from its neighbors and only reacts to changes caused by movements in a scene. Data are transmitted immediately and are scene driven, resulting in a stream of events with a microsecond time precision equivalent to conventional high-speed vision sensors, with the addition of being low power and sparse [101]. This type of acquisition increases the sensor dynamic range and reduces power computation.

Spiking Neural Networks (SNNs [40]) are computational models using neural stimulation. It has been shown that such networks are able to solve constraint satisfaction problems [15, 86], depth extraction from stereovision [94, 30] or flow computation [41, 45]. As they are mimicking real neurons behavior, they allow a massively parallel, low power calculation, which is highly suitable for embedded computation. The use of a SNN in this work is a natural choice to build a complete neuromorphic event-based system, from the signal acquisition to the final output of the depth information. This is advantageous because of the resulting low-power system promised by the spiking/neuromorphic technology. The developed architecture is particularly adapted on a variety of existing neuromorphic spiking chips such as the SpiNNaker [35], TrueNorth [82] or LOIHI [27] neural chips. More specific neuromorphic hardware, such as the 256 neurons ROLLS chip [104], can also be used. When combined with an event-based camera, power as low as 100mW is proven to be sufficient to achieve a realtime optical flow computation [45]. We are showing with this work that a low-power ($\leqslant$100mW), com-

putationally inexpensive and realtime DFD system can be similarly achieved.

Among the multitude of techniques developed by vision scientists to estimate depth, those called *depth from focus* (DFF) or *depth from defocus* (DFD) have the great advantage of requiring only a monocular camera [36]. The DFF method uses many images, and depth clues are obtained from the sharpness at each pixel. This method is computationally expensive and the amount of data to process is substantial. On the other hand, DFD estimates the variance of spatially varying blur spots based on a physical model. This technique requires less images but at the cost of a greater error in positioning. Current methods that use DFD or DFF generate depth maps for static scenes only [119] as they are limited by the frame rate of the camera driven at maximum of 25 fps. The computer vision and engineering community have described a number of algorithms for defocus computation [131, 95, 84]. However, they typically require multiple concurrent images [141, 132, 96], lightfield systems [122], specific lens apertures [141, 63], correlations[75], specific hardware[74] or light with known patterns projected onto the environment [96]. The use of images and luminance implies high computational costs of around 17ms to process a single frame [75].

These approaches cannot serve as conceivable models of defocus estimation in natural visual systems, as mammalian usually operate on a complete different data format and acquisition principles. Early studies [76, 78] show that the border between blurred and sharp regions can be used to establish the depth-order of objects. For example, an out-of-focus target with a blurry textured region and a blurry border was perceived to be located proximal to the plane of focus, while an out-of-focus target with a blurry region and a sharp border was perceived to be located distant to the plane of focus. Recent findings in neurosciences show that blur perception in human is a dynamic process that allows depth assessment. In particular, the retinal defocus blur provides information regarding the relative and/or absolute distance of objects in the visual field [43]. Recently[89], it has been demonstrated that subjects were able to detect the relative distance of two vertical edges, justifying that the retinal blur allowed the subjects to judge target distance differentially without any other depth cues. Other studies demonstrated that motor efference and/or sensory feedback related to the blur-driven accommodative response contain sufficient information to estimate the absolute distance of visual targets [33]. In addition, information derived from image blur can be integrated by the visual system with other visual cues (e.g., retinal disparity, size, interposition, etc.), which would assist in enabling one to judge the depth order of objects over a range of distances [22, 23, 77, 78, 80]. The addition of blur information can improve the speed and accuracy in such a depth-ordering task [79].

2.2 MATERIALS AND METHODS

2.2.1 Event based cameras

Biomimetic neuromoprhic silicon event-based cameras are a novel type of vision sensor that are data driven. Unlike their frame-based counterparts, they are not controlled by artificially created timing and control signals (frame clock) with no relation to the source of the visual input. Events are generated when significant changes of the relative luminance occur at the pixel level as shown on Figure 2. The visual output is in the form of an address event (AER) and encodes the visual information in the time dimension at the microsecond time precision. As soon as a change of luminance is detected, the process of communicating the event off-chip is initiated. The process executes with low latency, of the order of a microsecond, ensuring that the time at which an event is read out from the camera inherently represents the time at which a contrast change is detected. Let $e(x, y, p, t)$ be an event occurring at time t at the spatial location $(x, y)^T$. A positive change of contrast will result in an "ON" event ($p = +1$) and a negative change of contrast will result in an "OFF" event ($p = -1$). The threshold n beyond which a change of contrast is high enough to trigger an event is tuned according to the scene. Smaller intensity fluctuations do not generate any event and are not recorded.
The camera used in our setup is an asynchronous sensor which has a 640×480 pixels resolution [101] with a high temporal resolution of $1\mu s$. This array of fully autonomous pixels combines both a luminance relative change detector circuit and a conditional exposure measurement block (not used in the paper). When no change of luminance is detected, no events are generated and the static information is not recorded. This reduces the data load and allows high speed online processing at the native resolution of the sensor.

2.2.2 Depth estimation

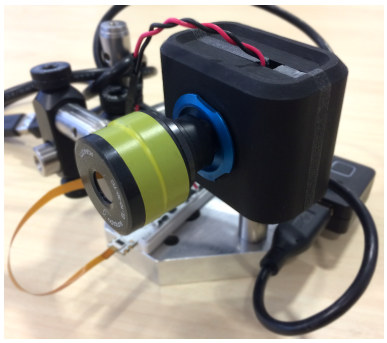
Geometrical optics relations

Let us consider a single thin convex lens of focal distance f with an infinite circular aperture. As shown on the Figure 3, the rays issued from a point $z = d$ converge to form an image behind the lens onto the detector according to the geometrical optic formula:

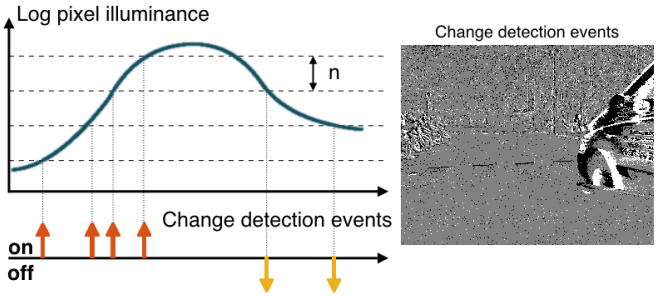
$$m \times m' = f^2 \tag{1}$$

In a stigmatic optical system, an object point is in focus when the image formed is a point as well. Any object out of focus forms a blurry spot on the detector. We can compute the distance of the camera to the objective as a function of the focal length of our optical system and the position of the object in focus:

$$D_{cam/obj} = f + m' = f + \frac{f^2}{m} = f + \frac{f^2}{d - f}. \tag{2}$$



(a)



(b)

Figure 2 – a) The neuromorphic silicon event based camera with the variable motorized focal lens controlled at 100Hz b) (left) Operating principle of event detection of an event-based camera: relative changes of the luminance greater than a predefined threshold n generate ON/OFF events when there is a positive/negative change of contrast. (right) Events output from the senors are shown as on the focal plane as a frame for purpose display, black dots represent OFF events while white dots represent ON events.

As this distance is fixed in our setup, increasing the focal value f results in an increase of the distance d at which an object is in focus and vice-versa. By tuning the focal value of the optical system, every plane of the 3D scene is successively in focus.

When an object is located before or after the focus point, it will form a blurry image of size s. Geometrical optic relations (Figure 3) give :

$$s = \frac{f^2}{N} \times \frac{|x - d|}{(d - f)z}, \tag{3}$$

with f the focal value of the optical system, N the numerical aperture, d the position of the object when in focus and $z(t)$ the variable position of the object over time. Due to the aberrations, diffraction phenomenon and non-idealities of the lenses, a Gaussian point spread function is commonly used to describe the defocus blur spot [64].

A real optical system has a finite aperture size, limited by the dimensions of the optics. The spatial resolution is also limited by the pixel size below which it is not possible to distinguish focus. This represents the circle of confusion, C, of the camera as shown on Figure 3. As a consequence, a range of several points will form an image "in focus" (the image spot size is

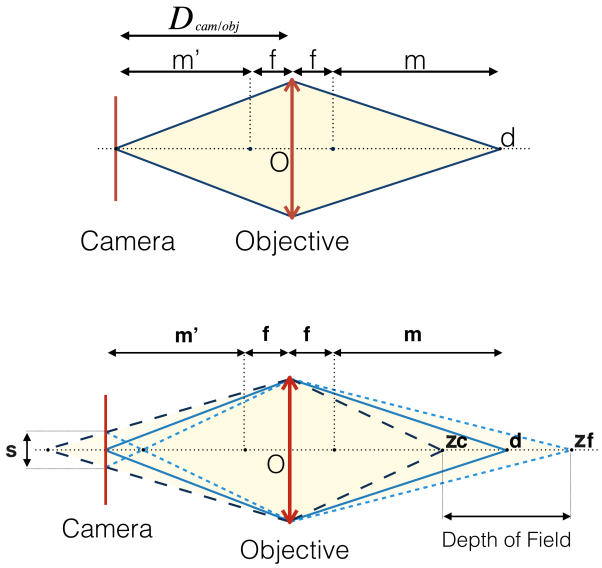


Figure 3 – (left) Stigmatic optical system. (right) Principle of depth from defocus.

smaller than the circle of confusion) on the detector for a given focal length. This range is called depth of field and the two limiting points are the values of z_{close} and z_{far} for which $s = C$, i.e.

$$z_{close/far} = \frac{d}{1 \pm \frac{CN(d-f)}{f^2}}, \tag{4}$$

The depth of field is then given by computing the difference between z_{close} and z_{far}:

$$D_oF = |zf - zc| = 2 \times \frac{CNd(d-f)}{f^2 - \frac{(CN(d-f))^2}{f^2}}. \tag{5}$$

The DoF of the lens is increasing with the distance of focus. Beyond a certain distance, called the hyper-focal, the whole scene appears in focus and differences in depth can no longer be distinguished. Ideally a DFD sensor should have an infinitely thin DoF for each focusing distance and an infinite hyper-focal. In practice one needs to minimize the DoF and increase the hyper-focal to have the best spatial resolution in depth on the longest distance possible.

Depth from the time of focus

The spread parameter $\sigma(t)$ is proportional to the diameter $s(t)$ of the ideal blur circle, i.e. $\sigma(t) = \alpha s(t)$. The resulting intensity onto the sensor, at a pixel (x_i, y_i) is:

$$I_{i,j}(x, y, t) = A.\exp\left(-\frac{r_i^2}{2\sigma(t)^2}\right). \tag{6}$$

with $r_i^2 = (x - x_i)^2 + (y - y_i)^2$ and A the amplitude. At the pixel level the evolution of the intensity will depend on how close to the camera the object is. As a function of time, the standard deviation in I can be used to determine the time t at which an event is triggered by the pixel, assuming σ is invertible i.e.:

$$t = \sigma^{-1}\left(\sqrt{\frac{r_i^2}{2\left(\log A - \log I_{i,j}(x, y, t)\right)}}\right) \tag{7}$$

We are dropping subscripts (i, j) for readability purpose as what we are describing is valid for any pixel. Hence, given the intensity at an arbitrary time t_0, if the variations of its log reach some threshold $\pm n$ (described in the previous section), then:

$$\log \frac{I(x, y, t)}{I(x, y, t_0)} = \pm n \text{ and } \log I(x, y, t) = \log I(x, y, t_0) \pm n. \tag{8}$$

This gives the time when an event is emitted according to (7):

$$t = \sigma^{-1}\left(\sqrt{\frac{r^2}{2(\log A - \log I_0 \mp n)}}\right) \tag{9}$$

The sign of n is chosen according to the polarity of the spiking event, itself related to the sign of the intensity's derivative:

$$\text{SIGN}(n) = \text{SIGN}(p) = \text{SIGN}\left(\frac{dI}{dt}\right) \tag{10}$$

When the derivative is positive the polarity will be +1 (ON event) and -1 when negative (OFF event). Eq(9) expresses when an event will be emitted w.r.t. n and to a reference event measured at t_0. As we reach focus, the value of σ will be constant for small duration of time, therefore the derivative of I, $\frac{\partial I}{\partial t}$ is equal to 0, followed by a polarity change as shown in Fig.4(c) and expressed in the temporal domain in Fig.4(d) around 50ms. The detection of focus can then be determined by detecting the time t_f of the polarity change that can be estimated from the average timing between the consecutive ON and OFF events. We can then estimate the size of the defocus blur $s(t_f)$ according to (7) and deduce from (**??**), the depth information $z(t_f)$ as:

$$z(t_f) = \frac{\mp df^2/N}{S(t_f)(d - f) \mp f^2/N}. \tag{11}$$

The change of sign in z corresponds to the focal length that is the closest to the focus. Parameters d and f are controls of the liquid lens device.

Defocus with an event-based camera

Let us consider a small luminous object that will successively be out of focus, in focus and out of focus again. When a sweep of the focal length over its dynamic range is carried out, objects will successively appear out of focus, then in focus and out of focus again. The blurry spot around the

object will therefore shrink until the object is sharp and grow again as shown in Fig.4(a) and in Fig.4(b) for a cross section of the blur spot. The size of the blur spot increases in connection to the distance respectively to the depth of field (DoF) location. When the object is in focus, the image spot will have its minimum size and the contrast will be maximum (sharp edges).

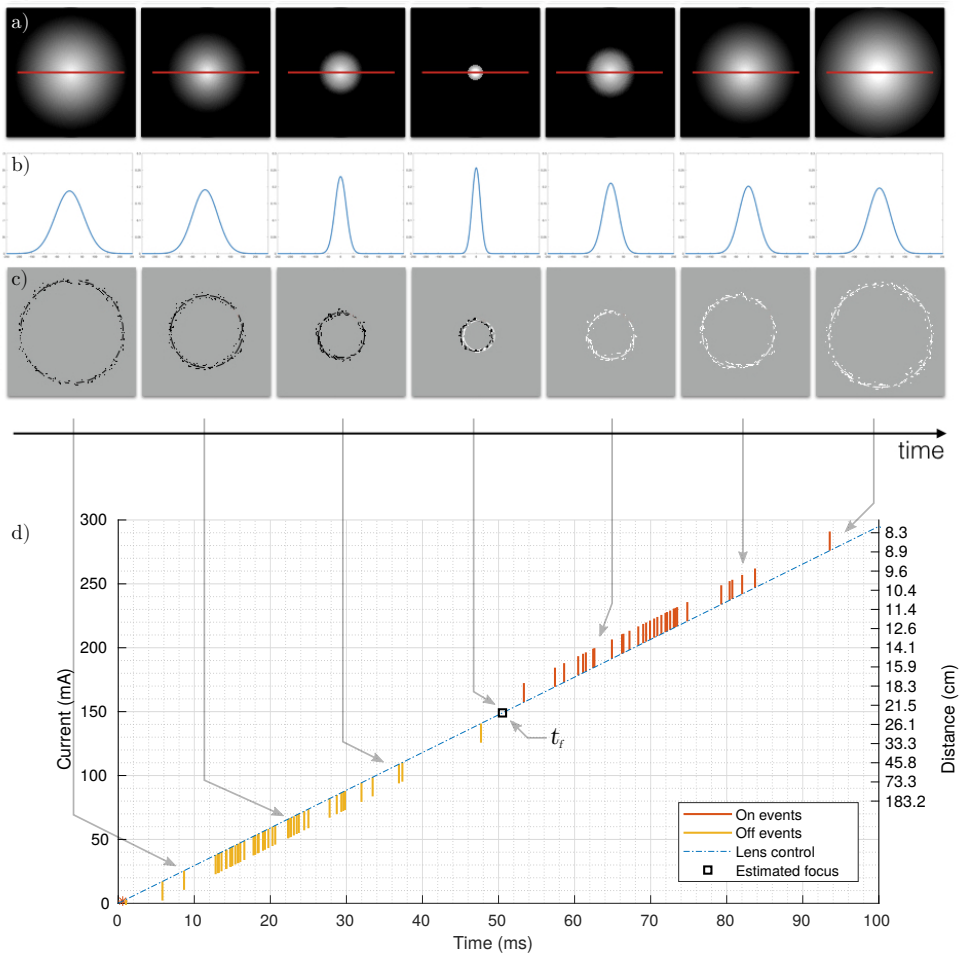


Figure 4 – (a) Successive snapshots of a sphere when sweeping the focus range. The red line represents a line of pixels in the y direction. (b) Variations of the intensity profile along the red y-axis on the above snapshots. (c) Events corresponding to the sweeping of the focus range, in black are OFF events and in white ON events. (d) Representation of spikes among a single pixel, according to the driving current of the liquid lens. Here, the focus point is estimated to be at 22.6cm from the sensor.

2.2.3 Liquid lens control

The optical system shown in Fig.2(a) is composed of three components:

— an electrically focus-tunable liquid lens with a 10 mm clear aperture and focus range f_{ll} ranging from 50 to 120 mm [16].

— an offset lens with a focal $f_o = -150$ mm. It acts as a relay imaging system between the focus-tunable lens and the objective and ensures a proper focus.

— an objective lens with focal length $f_{ol} = 35$ mm, $f_{ol}/2$ objective lens. This objective is a good compromise between large focal value, large clear aperture and low bulk (23.4mm length). It is used to form an image directly on the camera pixel array.

More details are given in Supplemental Data.

The thin lens approximation is given as follows:

$$d = f_{eq} + \frac{f_{eq}^2}{D_{cam/obj} - f_{eq}}, \tag{12}$$

where d is the position of the point in focus, f_{eq} is the global optical system's equivalent focal value (liquid lens + offset lens + objective lens) and $D_{cam/obj}$ is the distance between the camera and the object. See Supplemental data for details.

The thin lens approximation assumes that the equivalent focal length f_{eq} of our optical system is :

$$\frac{1}{f_{eq}} = \frac{1}{f_{ll}} + \frac{1}{f_{ol}} + \frac{1}{f_o} \tag{13}$$

with f_{ol} and f_o being respectively the objective lenses and the offset focal lengths.

2.2.4 Spiking neural network

To estimate t_f for each pixel, we are looking for the smallest time interval between two consecutive events of opposite signs. We implement a Spiking Neural Network (Figure 5.a) based on Leaky Integrate and Fire neurons ([60]) to process the spikes from the output of the neuromorphic silicon retina. For every pixel, five neurons are required. Figure 5.a shows events generated by a circle going in and out of focus. At time t_1, the stimulus in front of the receptive field generates a ON event (orange - Figure 5.c). The synaptic weight between the ON and B_{on} neurons is not strong enough to trigger yet the B_{on} neuron (Figure 5.d). As a second spike is generated by the same neuron at time t_2, the B_{on} neuron reaches its threshold value and spikes (Figure 5.d). A small inhibition link to the OUT neuron ensures that the OUT neuron won't fire now. After the focus, at time t_3, we have a polarity inversion : the OFF neuron fires, thus exciting the output neuron that fires (Figure 5.e). The next OFF spike, at time t_4, activates the B_{off} neuron, thus preventing the OUT neuron to fire again in response to the future OFF spikes. Finally, the Sync neuron is triggered by the liquid lens, warning that the sweep is over and reseting the OUT neuron to its initial state. The depth can then be extracted as the timing between the OUT and Sync spikes.

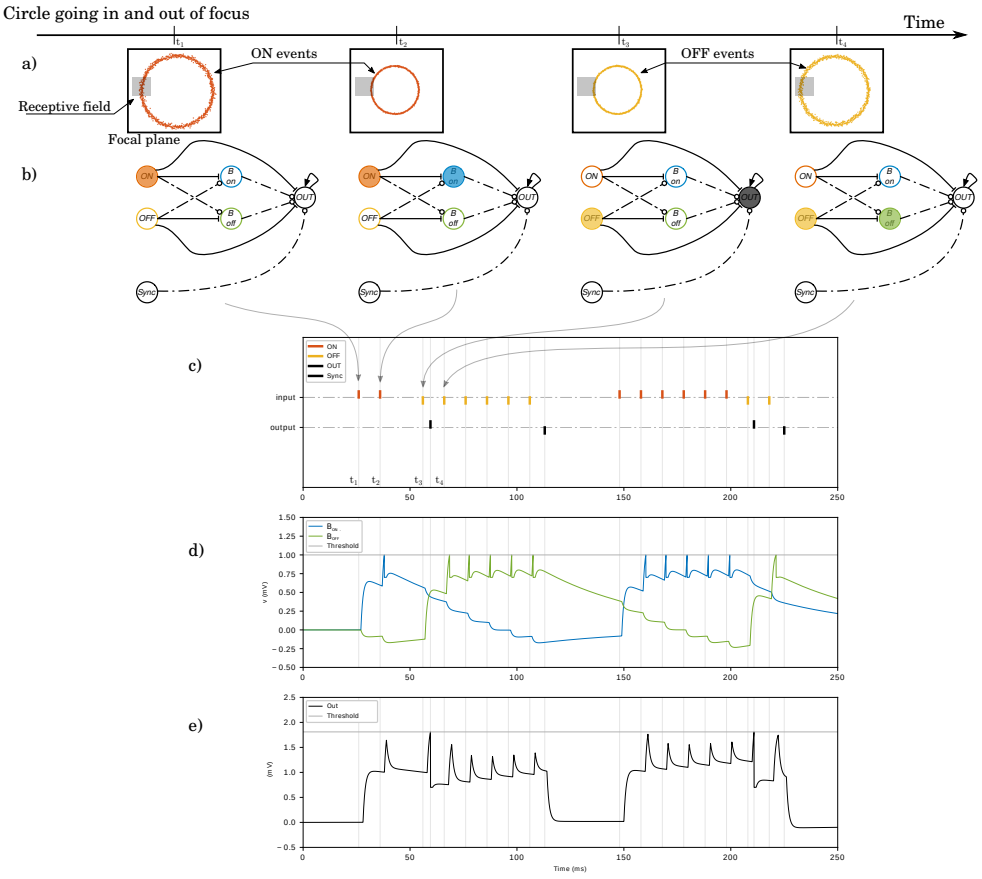


Figure 5 – Spiking neural network. a) Input data : a circle going in and out of focus, in front of a receptive field (a single pixel) b) Neural network for focus detection composed of two input neurons, *ON* and *OFF*. They directly connect to the output neuron, and also to two blocker neurons B_{on} *and* B_{off} that are inserted to avoid parasite firings of the output neuron due to a sequence of only ON or OFF polarity events. A synchronization with the liquid lens via the *Sync* neuron is added, in order to encode the depth in the length of the spike train. c-e) Simulation of the SNN with NEST. (c) The input spikes (ON and OFF events) and the output of the network (OUT and Sync). The point of focus is given by the OUT neuron, while the distance is encoded in the timing between the OUT and SYNC spikes. (d) Membrane potential for the two blockers neurons. After the first spike of its respective polarity, the blockers send a inhibition to the output neuron. (e) Membrane potential of the output neuron. Spikes from the same polarity do not allow the output neuron to reach its firing threshold, while a succession of ON and OFF events make the output neuron fire.

2.3 RESULTS

Results are obtained for a field of view of 15° and a depth that ranges from 0.12 to 5.5 m. The distance upper bound corresponds to the hyper-focal distance of the global optical setup. The sparse nature of the data allows the algorithm to operate in real time at the native resolution of the

sensor ($640 \times 480 pixels$).

The Spiking Neural Network previously described in section 2.2.4 was implemented using the PyNN framework [28] and simulated using the NEST neural simulator [39]. All neurons are modeled as Leaky Integrate-and-Fire (LIF) neurons. Results are presented on Fig.5. We set the dimension of the network to fit a region of 447×447 pixels, the network then using 999045 neurons. This amout is compatible with existing neuromorphic hardware implementation on the TrueNorth platform (1 million neuron [82]) or SpiNNaker capability [35].

To better understand the possibilities and limits of the system, we performed a simulation on synthetic data generated with a controlled optical setup where all parameters can be tuned. The aim of this simulation is to study the algorithm without constraints from the physical setup. Fig.6 shows three snapshots of the events generated during a sweep of a car. Fig.6-d) shows the reconstructed depth computed by the system.

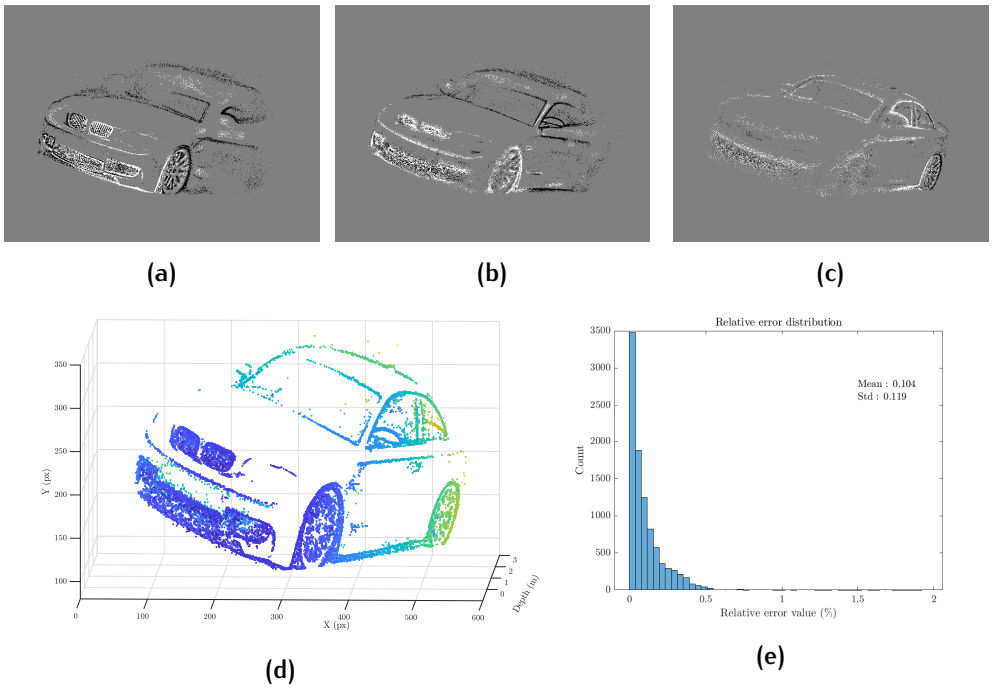


Figure 6 – (a)-(c) Snapshots during a sweep of an object (d) Reconstructed depth scene for the car. The depth is also color-coded for clarity. (e) Distribution of the error. The mean relative error is at around 0.1% and a standard deviation of 0.12%.

All the parameters being known we can estimate the relative error to the ground truth. We notice that most of the error is located at the front of the car on the grating where repetitive patterns are located. This is a known limitation of several vision algorithms such as stereo matching, which will be further discussed in section 2.3.1. Fig.6 (e) displays the error repartition with a mean relative error of 10.4%. An example video on a car is available online[44].

The second experiment, the depth estimated from the DFD is assessed with our setup on a monitored scene where the ground truth is provided by a Microsoft Kinect sensor. The Kinect is taken as the reference similarly to previous studies [57][69], reporting reconstruction precisions of few mm at 50cm to 3cm at 3m. Fig.7 shows the setup and the depthmap computed for the presented neuromorphic technique with a comparison with the groundtruth depthmap: the error is increasing relative to depth. Up to 1.5m, the relative error is upper-bounded at 4% and increased up to 23% at 2m. This is however an expected result as the optical system's focal length is reaching the hyper-focal.

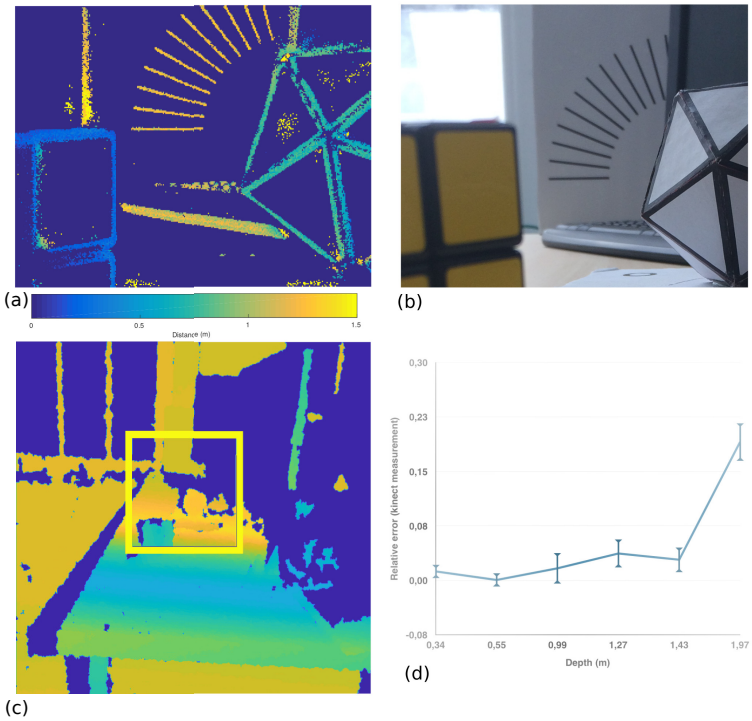


Figure 7 – **(a)** Depth map from the developed setup (raw data, no post-processing) **(b)** Conventional Image of scene for display purposes **(c)** Depth map from the Kinect used as reference. The yellow square corresponds to the field of view. **(d)** Relative error for this scene compared to Microsoft Kinect. The relative error increases with depth.

The third experiment shows reconstruction for several objects with different textures and sizes. Fig.8 shows for each object its corresponding depth map while the lens is sweeping through the object.

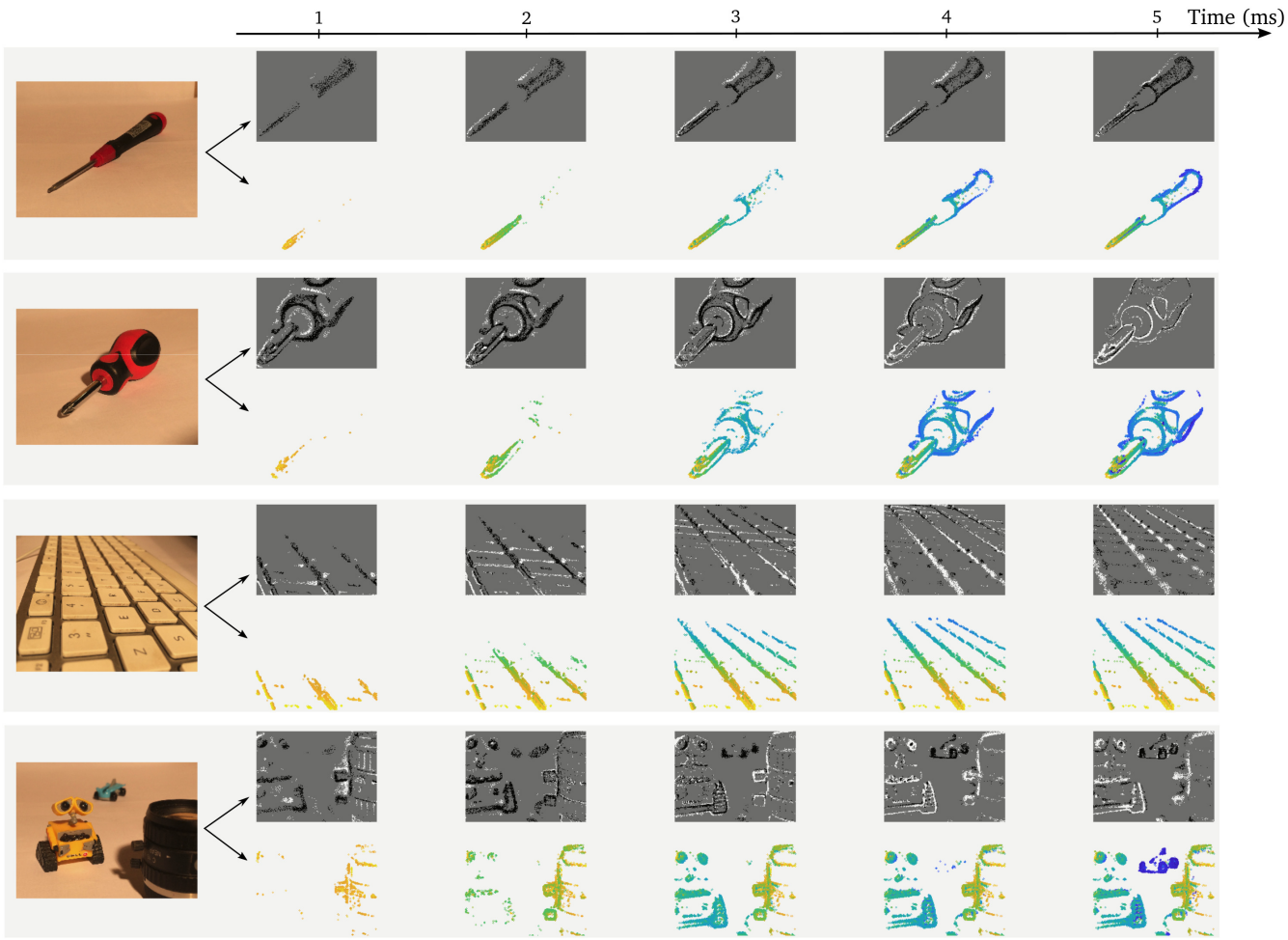


Figure 8 – Snapshots of the event stream, and associated depth maps during a sweep (5ms) for multiple objects. Black and white dots are the OFF and ON events from the event-based silicon retina, as described in Section 2.2.1. Distance is color-coded.

2.3.1 Remarks and limitations

The algorithm is inexpensive in computational power, in the presented experiments it is able to deal with more an average of 15 million events per second. A shaky scene viewed by the event-based camera will generate at most 2 million events per second. In the worst case, thus the current approach is 7 times faster than real time. However for most objects used it is more around 20 times faster than real time using an off-the-shelf laptop. This algorithm can be easily embedded on portable devices such as smartphones or autonomous vehicles as an ideal method for low power solutions to obstacle side-stepping or 3D scanners.

As pointed out during experiments, repetitive patterns can lead to incorrect depth estimation. Fig.9 shows this situation for simulated data. If we consider two objects that are well separated (Fig.9.f), the sweep of the liquid lens will produce an event stream (Fig.9.l) with non overlapping spikes. Fig.9.j is a snapshot of the sweeps' beginning. The four OFF edges are distinct. As the focus evolves, we reach the focus point for object 1 (Fig.9.i). The two edges O_{1L} and O_{1R} of object 1 now generate ON events. After the focus point for object 2 (Fig.9.h), the two other edges O_{2L} and O_{2R} now generate ON events. As the objects are in a sufficient relative distance, the edges O_{1R} and O_{2L} are not overlapping.
If we consider two objects that are close each other (Fig.9.a), the sweep of the liquid lens will now produce the event stream shown in Fig.9.k. As the sweep starts (Fig.9.e), only the external edges of objects 1 and 2 (O_{1L} and O_{2R}) generate OFF spikes. As the focus reaches object 1, object 1 generates ON spikes and object 2 OFF spikes. The two middle edges (O_{1R} and O_{2L}) are now superimposed, with two different polarities, causing the failure of the algorithm (Fig.9.d).

2.4 CONCLUSIONS AND DISCUSSIONS

In this paper we proposed a spiking neural network model that solves the depth from focus efficiently by exploiting an event-based representation amenable to neuromorphic hardware implementations. The network operates on visual data in the form of asynchronous events produced by a neuromorphic silicon retina. It processes these address-events in a data-driven manner using artificial spiking neurons computation units. This work introduces a valid explanation and a robust solution to depth estimation from defocus that has not been reported in the literature. The overall system matches recent existing literature of neuroscience, biological retinas and psychophysics studies on the role of defocus in the visual system. This network is nonetheless an abstract simplification of the depth estimation problem that must surely combine more complex information in biological systems. More importantly, this study should be coined depth from focus rather than from defocus as the neural structure developed aims at detecting the exact time of focus during a sweep.
Although five decades of research tried to solve the problem of depth from defocus, the fundamental difference and novelty of this work is that the net-

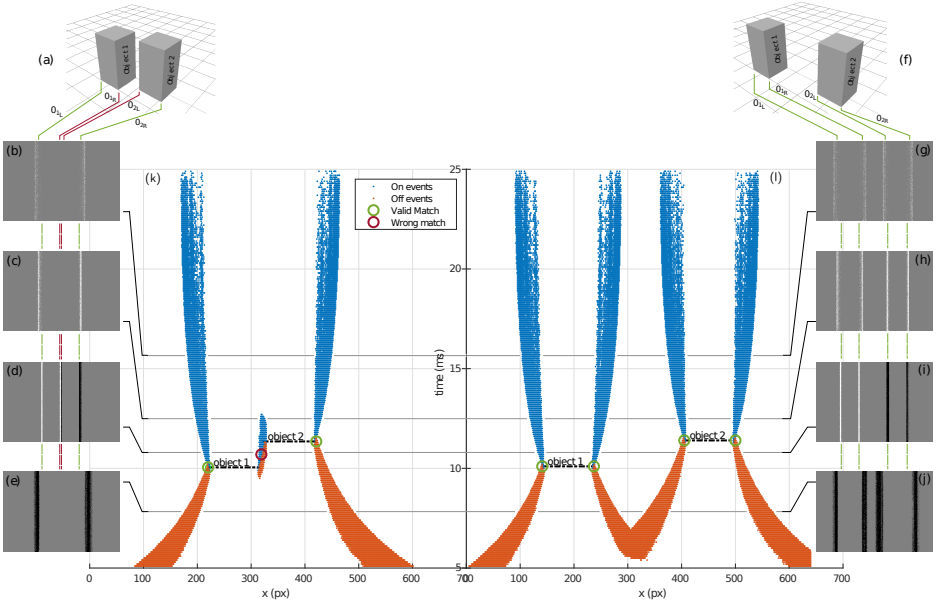


Figure 9 – Highlighting of the wrong depth measurements for two closeby edges. The two central plots show events in the x-time plane, smashing the y-dimension. Events are color coded with their polarity (red for OFF events, blue for ON events). The right one is a valid case, with no overlap. The left one contains an overlap in the event stream, leading to wrong depth deductions in this case. 4 snapshots of events are presented for every case

work proposed operates using exclusively precisely-timed temporal contrast events. These events are measured directly from the neuromorphic silicon retina, which models only the transient responses of retinal cells (i.e., of the Y-ganglion cells), without including the sustained ones, yet present in the system. While the sustained information is present in the silicon retina used, we show that this information is not necessary to provide depth estimation from defocus. Silicon retina transient responses produce single events. Their precise timing plays a crucial role in the estimation of blur and more importantly in determining when the observed object is in focus.
In contrast, the vast majority of computational models of depth from defocus are based on images that are known to be absent from the visual system and only rely on luminance information. Additionally, none of them use the precise timing of spikes. In these models, convolutions techniques are used to determine the level of blur. These methods are computationally expensive and meaningfully slower as several acquisitions are often needed to provide an accurate result. By contrast, the model we presented does not incorporate any notion of filtering or convolutions. These choices are based on the perception of spatial contrast, whereas the presented model solely responds to temporal contrast.

Whether the brain is using such a technique to estimate depth from defocus is an open question. However due to the nature of precisely timed information output by biological retinas [120] convolutions algorithms cannot provide a viable explanation as the stroboscopic nature of image acquisition and luminance use is incompatible with neural systems. Instead, we

show that the change of polarity at the pixel level contains sufficient information to estimate depth from defocus. Recent findings in physiology show that several mechanisms used by our methodology exist in the Nature. Biological retinas contain several types of ganglion cells, each informing the brain about a particular content of the visual scene, such as motion, edges or chromatic content. In a recent paper, a newly discovered ganglion cell type 'On-delayed' is described [72]. This cell has been shown to respond vigorously to increasing blur. Its degree of firing directly encodes the amount of high spatial frequencies contained in its receptive field. More importantly, this cell gets input from both ON and OFF polarities. While it is currently unknown how this defocus information is used by the brain, it is most likely that this information projects to the visual thalamus and cortex and also to midbrain structures where accommodation is controlled [4].

We expect the most significant impact of our model to be in the field of artificial vision. Today's machine vision processing systems face severe limitations imposed both by the conventional sensors front-ends (which produce very large amounts of data with fixed sampled frame-rates), and the classical von Neumann computing architectures (which are a ected by the memory bottleneck and require high power and high bandwidths to process continuous streams of images). The emerging field of neuromorphic engineering has produced efficient event-based sensors, that produce low-bandwidth data in continuous time, and powerful parallel computing architectures, that have co-localized memory and computation and can carry out low-latency event-based processing. This technology promises to solve many of the problems associated with conventional computer vision systems. However, the progress so far has been chiefly technological, whereas related development of event-based models and signal processing algorithms has been comparatively lacking (with a few notable exceptions). This work elaborates on an innovative model that can fully exploit the features of event-based visual sensors. In addtion, the model can be directly mapped onto existing neuromorphic processing architectures. Results show that the full potential is leveraged when single neurons from the neural network are individually emulated in parallel. In order to emulate the full-scale network, however, effcient neuromorphic hardware device capable of emulating large-scale neural networks are required. The developed architecture requires few neurons per pixel and is implementable on a variety of existing neuromorphic spiking chips such as the SpiNNaker [35], TrueNorth [82] or LOIHI [27] neural chips.

3 SPARSE CODING

This chapter introduces an unsupervised compact architecture that can extract features and classify the contents of dynamic scenes from the temporal output of a neuromorphic asynchronous event-based camera. Event-based cameras are clock-less sensors where each pixel asynchronously reports intensity changes encoded in time at the microsecond precision. While this technology is gaining more attention, there is still a lack of methodology and understanding of their temporal properties. This chapter introduces an unsupervised time-oriented event-based machine learning algorithm building on the concept of hierarchy of temporal descriptors called time surfaces. In this work we show that the use of sparse coding allows for a very compact yet efficient time-based machine learning that lowers both the computational cost and memory need. We show that we can represent visual scene temporal dynamics with a finite set of elementary time surfaces while providing similar recognition rates as an uncompressed version by storing the most representative time surfaces using clustering techniques. Experiments will illustrate the main optimizations and trade-offs to consider when implementing the method for online continuous vs. offline learning. We report results on the same previously published 36 class character recognition task and a 4 class canonical dynamic card pip task, achieving 100% accuracy on each.

3.1 INTRODUCTION

Neuromorphic event-driven time-based vision sensors operate on a very different principle than conventional frame-based cameras. Instead of acquiring static images of a scene, these sensors asynchronously record pixel intensity changes with a high temporal precision (around 1μs). The event format differs significantly from frames, and therefore conventional machine learning algorithms cannot be directly applied if one wants to fully use its potential and temporal properties in terms of power consumption, computational cost and low memory requirements. Previous notable work on object recognition using event-driven time-based vision sensors include real-time event-driven visual pattern recognition that recognizes and tracks circles of different sizes using a hierarchical spiking network running on custom hardware [114], a card pip recognition task on FPGAs, implementing different hierarchical spiking models inspired by Convolutional Neural Networks (CNNs) [98] and new methods such as HFirst [93] and recently HOTS [59] , an unsupervised algorithm which fully considers the spatio-temporal aspects of event-based sensors. In this chapter, we introduce a compact hierarchical event-driven multi-temporal framework to learn spatiotemporal patterns of dynamic scenes extending the concept of hierarchically increasing spatiotemporal-scales time-surfaces introduced in [59] . A time-surface

is a descriptor that provides a time context around an incoming event and describes the temporal activity in its surrounding. This descriptor, applied in a multilayer architecture requires a high number of features to correctly characterize a dynamic visual input and therefore high memory and computational costs. Ideally there must be a one to one relation between interesting spatio-temporal time-surfaces detected in the scene and the ones stored in each layer of the architecture. The aim of this work is to present a new formulation of the method in order to reduce the number of features using sparse coding to express any time-surface as a linear combination of elementary time-surfaces rather than selecting the most representative ones using clustering techniques (iterative K-means [6] in the original paper).

3.2 EVENT-BASED CAMERAS

Biomimetic event-based cameras are a novel type of vision sensor that are event driven. Unlike their frame-based counterparts, they are not controlled by artificially created timing and control signals (frame clock) with no relation to the source of the visual information. Events are generated when significant changes of the relative luminance occur at the pixel level as shown in Figure.10. The visual output is in the form of an Address Event Representation and encodes the visual information in the time dimension at the microsecond time precision. As soon as a change of luminance is detected, the process of communicating the event off chip is initiated. The process executes with low latency, of the order of a microsecond, ensuring that the time at which an event is read out from the camera inherently represents the time at which a contrast change was detected. The camera used in our setup is an Asynchronous Time-based Image Sensor (ATIS) [100] which has a 304×240 pixel resolution. This array of fully autonomous pixels combines both a relative luminance change detector circuit and a conditional exposure measurement block. Only the change detector circuit will be used for our experiments. When no change of luminance is detected, no events are generated and the static information is not recorded. This reduces the data load and allows for high speed online processing.

3.3 METHODS

In this section, we will introduce the notion of time-surface, describing the spatio-temporal surrounding of an event, before showing that these so defined time-surfaces can be clustered using a sparse coding algorithm, thus allowing a drastic reduction of the total number of features used for classification, which directly impacts the hardware implementability of the algorithm, as discussed in next section.

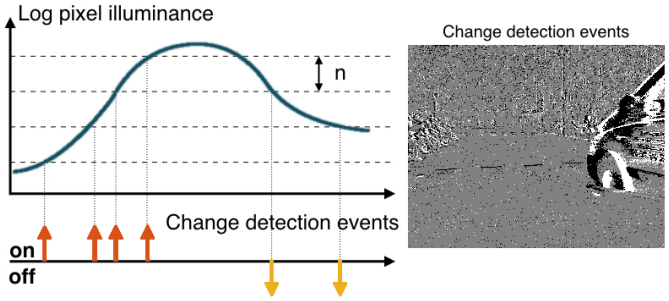


Figure 10 – a) The ATIS sensor. b) Functional diagram of an ATIS pixel. Two types of asynchronous events, encoding change and brightness information, are generated and transmitted individually by each pixel in the imaging array. Each time the luminance of the considered pixels rises a level (a relative change from previous measurement), a change detection event is generated, like at time t_0. This triggers a second circuitry which measures the absolute gray level value. This value is coded in the timing between two events occurring at t_1 and t_2. This timing encodes the time required to capture a given amount of light, and is then inversely proportional to the absolute gray level.

3.3.1 Time-surface construction

We consider a stream of visual events (Figure 11.b), which we define as $ev_i = \{x_i, y_i, t_i, p_i\}$ where ev_i is the i-th event, and consists of its spatial location (x_i, y_i), its timestamp t_i, and its polarity p_i, with $p_i \in$ {-1, 1}, for respectively a decrease or increase in luminance.
In [59] the notion of time-surface feature S_i is introduced. It represents the previous activity around the spatial location of an incoming event ev_i. Thus, for an incoming event ev_i, we define a time context (Figure 11.d) as an array of the last activation time in the $(2R+1)^2$ neighborhood (of radius R), centered at (x_i, y_i).
The time-surface S_i (Figure 11.f) around an incoming event ev_i is then obtained by applying an exponential decay (Figure 11.e) to each element of the time context [59]) .
For an incoming event ev_i at location (x_i,y_i), the value of its associated time-surface is:

$$S_i(u,v) = e^{-(t_i - t_{u,v})/\tau}$$

where $u \in [\![x_i - R, x_i + R]\!]$, $v \in [\![y_i - R, y_i + R]\!]$, $t_{u,v}$ the timestamp of the most recent event that occurred at the respective pixel, t_i the timestamp of the current event and τ the time constant of the exponential decay. As all the events are processed one after the other, the time difference $t_i - t_{u,v}$ is necessary positive. Figure 11.b shows this full process of time-surface generation.

3.3.2 Training phase: Finding the patch of projection basis

Following the sparse coding algorithm introduced in [90] , assuming that any time-surface $S(x, y)$ can be approximated as a linear combination of N

functions $\phi_j(x, y)$ for $j \in [\![1, N]\!]$ giving its estimation $\tilde{S}$ (for clarity, we omit the i subscript for each event) :

$$\tilde{S}(x, y) = \sum_{j=1}^{N} a_j \phi_j(x, y)$$

where the a_j are real linear coefficients ($a_j \in \mathbb{R}$ for $j \in [\![1, N]\!]$). This relation is equivalent to the projection of a time-surface S onto a subspace defined by the elementary time-surfaces ϕ_j. The feature estimation can be classically formulated as an optimization problem. The solution is given by minimizing the following residual error function E, set as the difference between the original surface and its reconstruction using the elementary time-surfaces linear summation (left member) and a measure of the sparseness of the coefficients (right member):

$$E = \underbrace{\sum_{x,y} \left[S(x, y) - \sum_{j=1}^{N} a_j \phi_j(x, y) \right]^2}_{\textit{reconstruction error}} + \underbrace{\lambda \sum_{j=1}^{N} |\frac{a_j}{\sigma}|}_{\textit{sparseness of the } (a_j)} \tag{14}$$

Where λ is a positive constant that determines the influence of the second term relative to the first, and σ a scaling constant.

The minimization is performed using a training dataset of events from the event-based camera [90] . Elementary time-surfaces are initialized with random uniformly distributed values between 0 and 1, while the coefficients values are initialized with the product between the initial features and the time-surfaces computed from the training dataset. We use the conjugate gradient descent method [103] to minimize the error E (i.e. maximize the similarity between S and $\tilde{S}$, by updating the coefficients a_j, while maximizing the sparseness of the coefficients).
Elementary time-surfaces are then updated by adding the residual error weighted by the coefficients obtained from the previous iteration of the minimization process:

$$\phi_j \longleftarrow E.a_j.\eta + \phi_j$$

where η is the learning rate. All time-surfaces are computed offline from the learning dataset. Then, they are fed to the minimization algorithm until convergence. The number of features is computed iteratively and set as the one providing the lowest reconstruction error on the training time-surfaces set.

3.3.3 Building a hierarchical model

In the initial architecture described in [59] , each layer has a set of elementary time-surface prototypes matching time surfaces from the observed scenes and learned during the training phase. When an input time-surface matches a prototype, an event is produced and transmitted to the next layer. A single input event will produce at most one output event. In the present

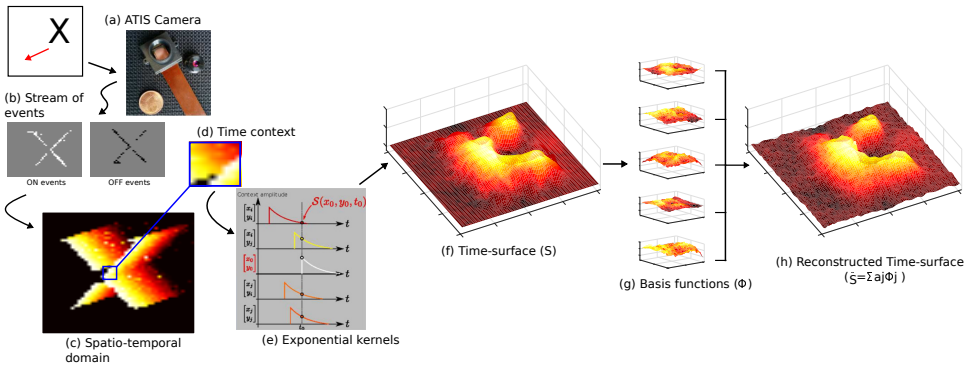


Figure 11 – Time-surface generation and decomposition over a basis. The ATIS camera (a) generates a stream of events (b) from an moving X character. The time context (d) of one event is extracted from the spatio-temporal domain (c) and convolved with and exponential decay (e), providing a time-surface (f). This time-surface is projected over a set of basis functions (g), which gives the resulting (h) reconstruction. Inspired from [59] .

model, a layer now contains a finite set of N elementary time-surfaces while the output of a layer represents the linear response of all the N elementary time-surfaces, represented by the projection coefficients of the input surface onto this basis.

Figure 14 shows the response obtained at the output of a layer, and how it is sent to the next layers in order to build the proposed hierarchical model. Figure 14.a shows an object moving in front of an ATIS camera. Each event is captured from the stream sent by the event-based camera. In Figure 14.b, time-surfaces for each event are built by a convolution with an exponential kernel of time constant τ_1, for a neighborhood of $(2R_1 + 1)$ side length. As described in section II, time-surfaces are sent to Layer 1, to find a projection basis of N_1 elementary time-surfaces shown in Figure 14.c. Once the elementary time-surfaces projection basis has been extracted, the learning process of Layer 1 is finished. Now each incoming time-surface is directly projected onto all elementary time-surfaces basis. The projection coefficients (Figure 14.e) are defined as the least square solutions that minimize the error function E (Equation 14). The a_j coefficients are constrained between -1 and 1. The projection coefficients are split into two groups (Figure 14.f-g), depending on whether their value is positive or negative, as detailed in [58] . For each feature j of the basis, a linear decay is applied to the projection coefficients as shown in Figure 14.g, and a new event ev_{out} is generated :

$$ev_{out} = \{x_{out}, y_{out}, t_{out}, p_{out}\}$$

where :

$$\begin{cases} (x_{out}, y_{out}) & = (x_{in}, y_{in}) \\ t_{out} & = t_{in} + \alpha\left(1 - |a_j|\right) \\ p_{out} & = j \quad | \quad j \in [\![1, N_1]\!] \end{cases} \tag{15}$$

were α is a timing scale factor describing the time span of the newly generated event-stream. The higher the coefficient, the higher the similarity

between the basis and the surface input. We then encode the similarity level into the time domain, as a delay of the output event depending on the similarity between two surfaces.
As shown in Figure 14.h, new events are convolved with an exponential decay of time constant τ_2, for a neighborhood of $(2R_2 + 1)$ side length, in order to generate new time-surfaces. In Figure 14.i, output delayed spikes, for positive coefficients, according to the results of convolutions with each elementary time-surfaces basis are sent to Layer $(2, +)$ for training. The goal is now to determine the N_2 elementary time-surfaces of Layer 2. As shown in Figure 14.j, steps (d) to (h) are repeated. Then, the same steps (g) and (h) are applied to Layer $(2, -)$ for negative coefficients.

Since the nature of the input and the output of our model is the same, events produced at the output of Layers $(2, +)$ and $(2, -)$ can be sent to the next layer, and the process from step (b) to (g) is repeated (Figure 14.l-k). Layers are trained consecutively one after the other.

As in [59] , the main idea of the architecture is to gradually increase the complexity of spatial and temporal information. Information is integrated over larger and larger time scales. Each layer L_i increases its number of elementary time-surfaces N_i, the neighborhood radius R_i, and the integration time constant τ_i:

$$\begin{cases} \tau_{i+1} & = K_\tau \tau_i \\ R_{i+1} & = K_R R_i \\ N_{i+1} & = K_N N_i \end{cases} \tag{16}$$

The network is only defined by a set of six parameters : the initial conditions (τ_0, R_0, N_0) and the evolution parameters (K_τ, K_R, K_N). Comparisons with [59] are provided in the experiments section.

3.3.4 Classification

The output of the last layer of the hierarchical structure is fed to a classifier for pattern classification. We use a simple classifier to show that the model provides sufficient discrimination without the need for complex classification methods. However, in case of larger databases, where a variety of descriptors are generated for the same class, the use of more advanced classification algorithms might become necessary. We compute an histogram from the output of the last layer of the hierarchical model that contains the total number of responses of each feature to the input pattern, independently from its spatial position. This is the signature of the observed object, and will be used for further comparisons. This method is the same as the one explained in [59] . Figure 13b shows an example of such an histogram.

Each learning example is presented to the model, in order to learn its signature. In a second step, during the testing phase, the signature of incoming patterns are computed and compared to the signatures of each learned examples. The closest one is then identified and finally, the recognized object is obtained by a majority vote from the results of all the last sub-layers.

We will use two types of distances between signatures : the Euclidean distance and the Bhattacharyya distance [14] . The histogram signature for

classification, which is a weak classifier, is here chosen in order to prove the robustness of the proposed descriptor. Stronger classifiers exist (SVM, adaboost methods, ...), but their use in this context is outside the focus of this chapter.

3.4 EXPERIMENTS

(a) (b)

Figure 12 – Representation of the datasets used for the experiments: (a) Letters and digit dataset: consisting of the 26 characters of the roman alphabet and the digits from 0 to 9. (b) Flipped cards dataset: consisting of the four suits of a deck of cards (club, diamond, heart and spade). Inspired from [59] .

3.4.1 Letters and digits dataset

The model has been used with the "Letters and Digit" dataset, provided by Orchard et. al [93] (Figure 12-a). It consists of a set of moving characters. The dataset used for the learning phase contains the representation of 26 characters from the roman alphabet and ten digits (A-Z, 0-9). The dataset used for the testing phase contains 12 representations of each character, in [A-Z] and [0-9]. The goal is to identify the character or the digit. For this experiment, we tested our method on a three layer architecture, we set empirically the number of elementary time-surfaces for each layer to be respectively: $N_1 = 6$, $N_2 = 9$, $N_3 = 12$.

The neighborhood radius is empirically set to $R = 2$ throughout the whole architecture, while the integration times for the exponential decays are set to : $\tau_1 = 10$ms, $\tau_2 = 15$ms and $\tau_3 = 20$ms. We were able to obtain 100% of accuracy in classification, both with the Euclidean and the Bhattacharyya distance.

3.4.2 Flipped card deck

The second dataset we used is the "Flipped card deck" dataset, provided by Linares-Barranco et. al [98] (Figure 12-b). It consists of a deck of cards whose corner is flipped in front an event-based camera so that only the suit symbol of the card is visible. The goal is to identify the cards' suit. The original dataset contains 40 samples, 10 from each suit. The learning and testing datasets were generated by randomly taking respectively 7 and 3 examples of each suit from the original dataset. For this experiment, we tested our method on a three layer architecture, with the same parameters

used for the letters and digits experiment. We were able to obtain a 100% accuracy. We then decreased the number of features for each layer, in order to see the impact of such a modifications. For $(N_1, N_2, N_3) = (3, 6, 9)$, the recognition rate drops to 83%, while decreasing the number of generated spikes by a factor of 2.5 (see Table 1).

Table 1 – Results and comparison with original HOTS model

Algorithm	This work			HOTS [59]	
Dataset	Cards		Digits	Cards	Digits
Number of Centers	3-6-9	6-9-12	6-9-12	8-16-32	8-16-32
Recognition Rate	83%	100%	100%	100%	100%
Number of spikes	7 599 450	18 500 730	74 440 535	52 410	513 383

3.5 CONCLUSION

This work exposes a new approach for learning spatiotemporal features from an event based camera. It allows a compact representation of information by reducing the number of prototypes in a hierarchical learning structure using sparse coding basis decomposition. This reduction induces an increase in the number of events generated by the system. Unlike the original model, we have shown that it is possible to reduce the number of prototypes without strongly impacting the precision. Our model increases the activity of each cell. This, however, is not a limitation since the available neuromorphic hardware can handle a large number of spikes (signal constraints) despite being limited to a small number of neurons (hardware constraints). Limitations are then shifted to maximal firing rate and bandwidth, not to mapping constraints due to a lack of available neurons. It is important to emphasize that we assumed a floating point precision, no consideration was given to precision limitations. Future work will focus on characterizing a loss of precision, both in number representation and in basis projection errors. Moreover, the addition of a refractory period to each prototype, in order to limit the firing rate of a single cell, will be studied. All of this could lead to a full hardware implementation.

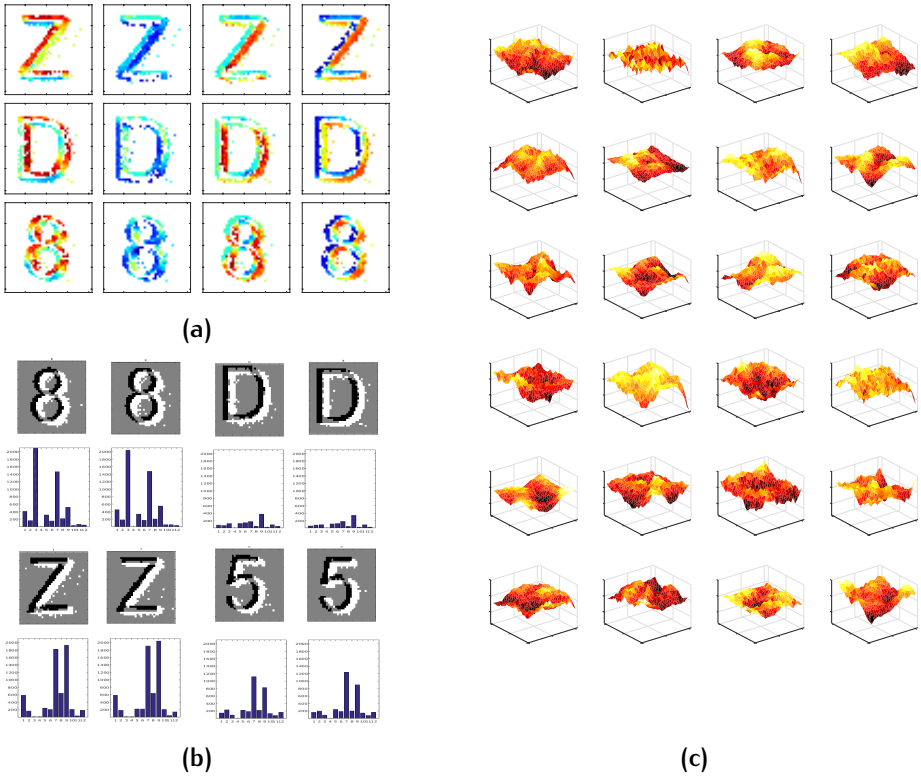


Figure 13 – Results for the character dataset. a) Output of the 3 layers network for three characters after learning. This output is color coded according to the responding basis (1 out of 24). We can see that some basis are specialized (red for horizontal edges, dark blue for curves, etc). b) Signature for two different representations of the same digit. The signatures are very similar between the same class and differ from class to class, allowing a good classification. c) 24 bases of the third layer of the model after learning on the letters and digits dataset. Because they are combination of responses of the previous layer, they are difficult to interpret as they are, but one can notice that they differ one from the other.

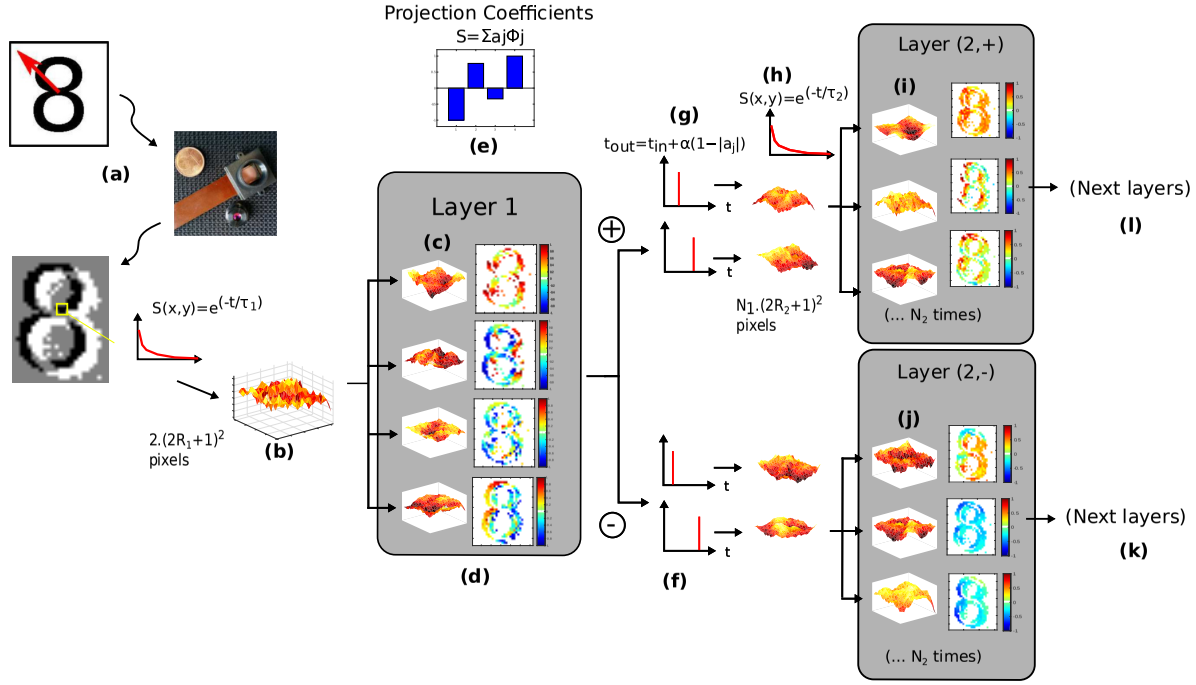


Figure 14 – Detailed view of a two layer architecture : **(a)** A moving object is presented to the ATIS camera which outputs a stream of events. **(b)** Each event ev_{in} is acquired and a time surface is computed using a τ_1 time constant. As the event stream here contains two polarities (ON and OFF), the time-surface contains the two polarities and is of size $2 \times (2R_1 + 1)^2$. **(c)** Time-surfaces are presented to layer 1 and an elementary set of N_1 prototypes *(cf. eq 16)* is extracted. **(d)** After learning this elementary set, recorded time-surfaces are presented again to Layer 1, and projected onto all prototypes. **(e)** The projection coefficients are obtained as the least square solutions that minimizes the error function E *(cf. eq 14)*. **(f,g)** All the projection coefficients are split between positive and negative values. The projection coefficients are convolved with a linear decay, and the results are used to determine the spike times t_{out} of the next emitted events ev_{out} *(cf. eq 15)*. **(h)** Timestamps are sorted in ascending order and convolved with an exponential decay of integration time τ_2 *(cf. eq 16)* to obtain new time-surfaces, of size $N_1 \times (2R_2 + 1)^2$. **(i)** Time-surfaces are presented to layer $(2, +)$**(i)** and $(2, -)$**(j)** and a projection basis of N_2 *(cf. eq 16)* elementary time-surfaces is estimated. **(k, l)** The same process can be repeated for the next layers, building in this way the proposed hierarchical model. Inspired from [59] .

4 SPIKE SORTING

With the increase of multi-electrode array size, spike sorting algorithm are often overwhelmed by the quantity of data to process. It is then impossible to have meaningful results in real-time for applications such as closed loop experiments. Furthermore, there is almost always the need of an external human operation to control and validate results. Here, we show that neuromorphic computation can yield to very good results in real-time unsupervised spike-sorting tasks. Our results demonstrate that considering the time as the most valuable information in signals helps extracting coherent information from noisy data. Comparison between the proposed method and state-of-the-art algorithms shows that event-driven computation allows to significantly reduce computation time while increasing efficiency, reaching up to 90% of recognition rate on real data. We anticipate this work to open new horizons for embeddable real-time devices for closed-loops applications and low-cost performance analysis of in-vivo data.

4.1 INTRODUCTION

State of the art spike sorting methods relies on prior knowledge of the dynamic of the signal for discriminant features extraction, and aren't suitable for real-time applications. Thus, these methods, regardless their efficiency, couldn't be considered as bio-inspired ones. This work aims to show that it is possible to extract, in real time, the dynamic of the signal. It has been shown that an event-based approach for pattern recognition [20][117][24][59][1] can yield to a better conceptualization of each new piece of information, with respect to past activity, both in spatial and temporal neighborhood. In this work, we will present a hierarchical event-based approach for spike sorting applications.

Spike sorting algorithms mainly rely on three steps : spike detection, feature extraction and classification. The first step often employs an automatic spike detection method (threshold detection in most of the case [112]). During a second step, a set of features is computed, *for example* using mixture of Gaussian kernels (Khadir et Al.[56]), wavelets transformation [105] or PCAs [112]. Then, the extracted features are assigned to cell types by learning algorithms, such as (un)supervised clustering.
In this chapter, I will first show a novel approach for features extraction, before explaining how it can be used for spike detection, allowing the whole pipeline to work in real-time.
The performances of the algorithm were tested both with a synthetic benchmark [137] and in-vivo recordings [49], and then compared to state-of-the-art algorithms [112, 56, 105].

4.2 METHODS

Here I detail how, from an analog signal, the proposed algorithm generates events, extracts features, clusters them in a hierarchical model, and then classifies unknown examples.

4.2.1 Model description

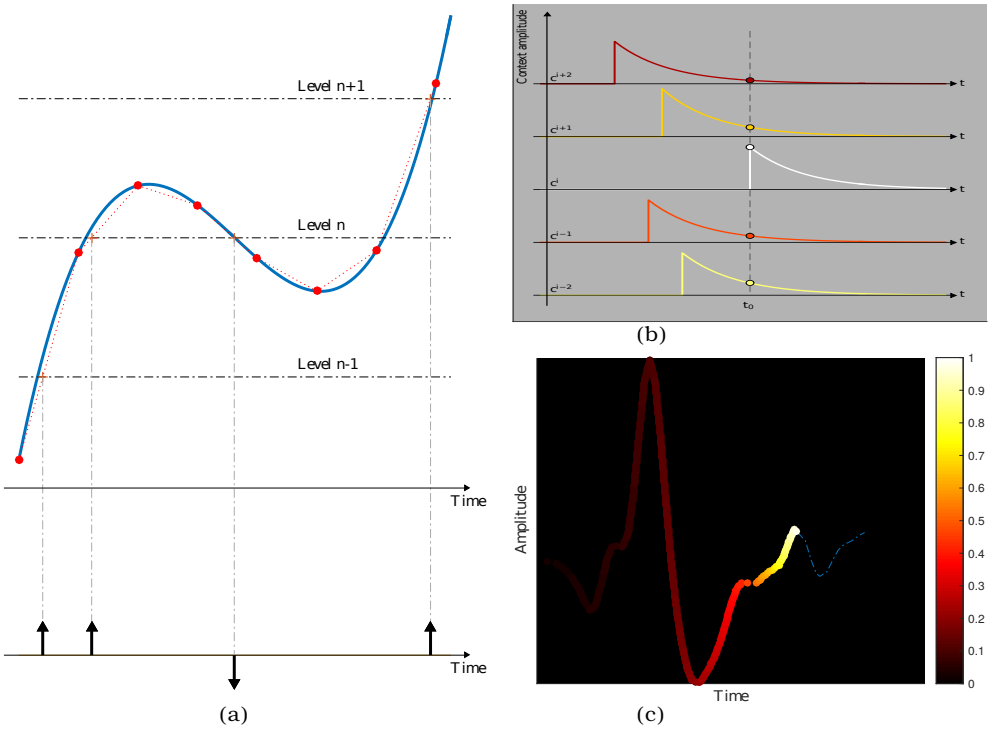


Figure 15 – Generation of a context. (a) Level crossing for spike generation. The sampled signal (red dots) is interpolated. Each time the linearly interpolated signal (dashed red line) crosses a level (horizontal dark dashed-dotted lines), a spike is generated, either positive or negative depending of the rising or falling behavior of the signal. (b) Generation of a context. When a spike occurs at channel c_i, all the values of the exponential decays among a spatial neighborhood are taken ($i \in [\![i-r, i+r]\!]$,r being the current radius value) and are our base features. (c) Exponential time decay among a spike. The value of the exponential kernel is color-coded : brighter pixels represent higher values than darker ones.

We will here give an overview of the proposed algorithm. Detailed explanations and equations can be found in the Method section.

When the input signal changes, our algorithm asynchronously generates events forming a spatio-temporal point cloud representing the signals' dynamical behavior. The sampled signal is first sent into a spike generator to (artificially) convert it in an event-based signal. As in silicon retinas [100], the variation of the signal triggers an event generation. In a more detailed approach, the sampled signal if first filtered by a high-pass, 3^{rd} order, Infinite Impulse Response (IIR) filter for DC filtering and low frequencies noise rejection. Then, the filtered signal is compared to a bank of levels. Each

time the signal crosses a level, a spike is triggered, both positive if the signal is increasing while crossing the level, or negative if the signal is decreasing (Figure 15-a). A linear interpolation between two samples is used for better time precision.
Each event $ev(i)$ can be seen as the following triplet :

$$ev(i) = \{t_i, c_i, p_i\}, i \in \mathbb{N} \tag{17}$$

where t_i is the timestamp of the event, c_i its channel (the crossed level) and p_i its polarity, $p_i \in \{-1, 1\}$ (*i.e.* positive or negative).

Considering a stream of events, we can associate for each event a description of its spatio-temporal neighborhood. This description will be referred as context, generated by convolving an exponential kernel (Figure 15-b) to the most recent activity on the surrounding of the incoming event (Figure 15-c). These contexts are then clustered using an online iterative clustering method [6]. Once this learning phase done (*i.e.* the centers are determined and represent the dynamics of the learned signal), we extract the response of the network for different spikes. Then, events from unknown spikes are fed in the network, and their signatures compared to the learned ones for classification.

Input data is presented to the network. Contexts are built with the convolution with an exponential kernel of time constant τ_1 and considering a spatial neighborhood of side length $(2R_1 + 1)$. Then, these contexts are clustered into N_1 centers. When a cluster is matched, an output event is produced, constituting the output of Layer 1. This output is of the same type as its input, as shown in Equation 17 and 24. Thus, the same process applied to Layer 1 can be applied to layer 2 using different parameters for space-time integration (τ_2, R_2, N_2).

As stated before, each layer l is only characterized by 3 parameters :
— τ_l, the time constant of the exponential kernel,
— R_l, the size of the neighborhood,
— N_l, the number of centers to be learned by the clustering algorithm.

The output of the last layer can be used as a feature for shape classification. The training of the recognition algorithm consists in two distinct steps. In the first one, learning data is presented in order to learn the centers computed as described in the previous section. Then, the same inputs are provided to the network and a histogram of centers activation (signatures) is built for each different class, using rather TSNE projection or offline KMeans clustering. These histograms represent the number of centers activated during the presentation of the example, independently of its spatial position, and are discriminant enough to allows proper classification.

4.2.2 Event generation

The sampled signal (48 kHz for the benchmark, 20 kHz for ex-vivo recordings) is first sent into a spike generator to (artificially) convert it in an event-based signal. Then, the sampled signal if filtered by a high-pass, 3^{rd} order, Infinite Impulse Response (IIR) filter for DC filtering and low frequencies noise rejection. Then, the filtered signal is compared to a bench of levels.

Each time the signal crosses a level, a spike is triggered, either positive if the signal is increasing while crossing the level, or negative if the signal is decreasing (Figure 15-a). A linear interpolation between two samples is used for better time precision.
Each event $ev(i)$ can be seen as the following triplet :

$$ev(i) = \{t_i, c_i, p_i\}, i \in \mathbb{N} \tag{18}$$

where t_i is the timestamp of the event, c_i its channel (the crossed level) and p_i its polarity, $p_i \in \{-1, 1\}$ (*i.e.* positive or negative).

4.2.3 Feature extraction and clustering

A spatio-temporal point cloud is formed with these events, representing the spike spatial distribution and dynamic behavior. Because this point cloud may contain information about the spikes dynamic and amplitude, we introduced the event context S_i of the event $ev(i)$ to convey information about surrounding activity just before time t_i.
Let $N(c_i)$ be a windowing neighborhood with length $2R + 1$ (in channels) around the event's channel c_i :

$$N(c_i)^c = \{c_i + c \quad | \quad abs(c) \leqslant R\} \tag{19}$$

Then, this event's context S_i is defined as :

$$S_i^c = exp\left(\frac{t_i - t_{ref}^c}{\tau}\right) \qquad for\{abs(c) \leqslant R\} \tag{20}$$

where t_{ref}^c is the last spike time at the c^{th} channel.
When a context has been computed, it is compared to a bank of context, or centers. The most-closely matching center will then generate an output event. First, we do have a set of N initial contexts $m_k, k \in [\![1, N]\!]$ where m_k takes the same form as S_i in Equation 20. These contexts are initialized with the first N incoming contexts. More formally,

$$m_k = S_k \qquad k \in [\![1, N]\!]$$

Then, they are clustered using an online Iterative Inverted Weight Kmeans (*IIWKmeans*, [6]). This clustering method was preferred to KNNs [26] or KMeans [68] because of its independence towards the initial conditions and its ability to work online. For each incoming context S, we define its nearest center m_{k^*} :

$$m_{k^*} = \underset{k}{argmin}(||S - m_k||) \tag{21}$$

where m_k is the k^{th} center. The update rule is then shown by:

$$m_{k^*}(t_s + 1) = m_{k^*} - \zeta a_{ik^*}(S - m_{k^*}) \tag{22}$$

with:

$$a_{ik^*} = -(n+1)\|x_i - m_{k^*}\|^{n-1} - n\|x_i - m_{k^*}\|^{n-2} \sum_{j \neq k^*} \|S - m_{k^*}\| \quad (23)$$

ζ and n are two learning parameters able to constrain the update rule.

Once the learning is over (*i.e.* all the centers converged), each context can be associated with a particular center m_k. So, the input stream of events is transformed into an output stream of centers' activation :

$$ev_{out} = [c_i, t_i, k_i] \quad (24)$$

where k_i is the index of the matching center m_{k^*}.

At this point, for noise (isolated context) rejection, it is possible to implement context rejection based on some thresholding among the distances and so prevent emitting a new event if the match between the context and the center was not strong enough.

4.2.4 Classification

The recognition by itself is done by comparing online signatures to the trained ones. The distance between two histograms (H_1, H_2) used here is the Bhattacharyya [14] distance :

$$d^b(H_1, H_2) = -\log \sum_i \sqrt{\frac{H_1(i)}{card(H_1)} \frac{H_2(i)}{card(H_2}}$$

4.3 RESULTS

4.3.1 Metrics

Noise level

The noise level n_l was defined as the reciprocal value to the signal-to-noise ratio (SNR) [137] :

$$n_l = SNR^{-1} = \left(\frac{A_{signal}}{A_{noise}}\right)^{-1}$$

where A_{signal} is the root mean square value from all the extracted spikes, and A_{noise} the one for the rest of the signal.

Performance rating

To compare our method to state of the art spike sorting algorithms, we used the Adjusted Mutual Information. This measure provides information about how good the clustering is. Unlike the recognition rate (trace of the confusion matrix divided by the sum of all the elements within the confusion matrix), this measure is adjusted by chance, meaning that a zero value stands for chance and a 1 value for a perfect classification. Details and properties can be found in [129].

Table 2 – Model parameters for the artificial benchmarking

	L_1	L_2	L_3	L_4	L_5
$\tau(\mu s)$	25	50	75	100	125
$R(px)$	2	4	8	16	32
N_c	4	11	39	147	562
ζ	5.10^{-5}	5.10^{-5}	5.10^{-5}	5.10^{-5}	5.10^{-5}

4.3.2 Benchmarking

The proposed algorithm was tested with a database of artificially generated data from simulating extracellular signals recorded with a single electrode. Nine real spikes where manually picked from extracellular tungsten micro-electrode recorded during Deep Brain Stimulation operation from the sub-thalamus nuclei. They are then corrupted with noise coming from a superimposition of activity of many distant neurons in the brain [137].

Results

In this section, we will present results of the proposed algorithm on the benchmarking database [137]. This database contains 32 sets of 9 artificially generated spikes, corrupted with noise. Figure reffig:voronoi-b shows the noise-free input data. The training of the algorithm was done with one full set of the intermediate noise-level. The learned centers where then used for building signatures and recognition for all the 31 remaining sets. The parameters used for this benchmark are summarized in Table 2.

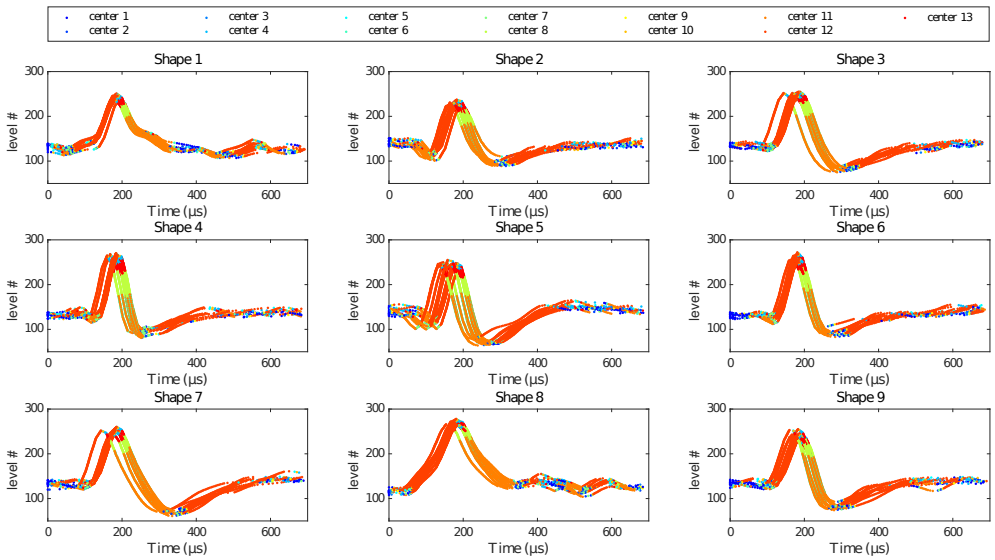


Figure 16 – Output of the last layer. For all the 9 different shapes, the matching clusters among a spike is color-coded. Our algorithm learns and extracts local curvatures that can be used for classification.

Figure 18-a shows the response among the centers of the fifth and last layer. The dynamic of the signal is well captured and clusters respond to local patterns of the input signal. Using high dimensionality reduction (*TSNE*, [67]) for data visualization, the projection of the signatures for some testing

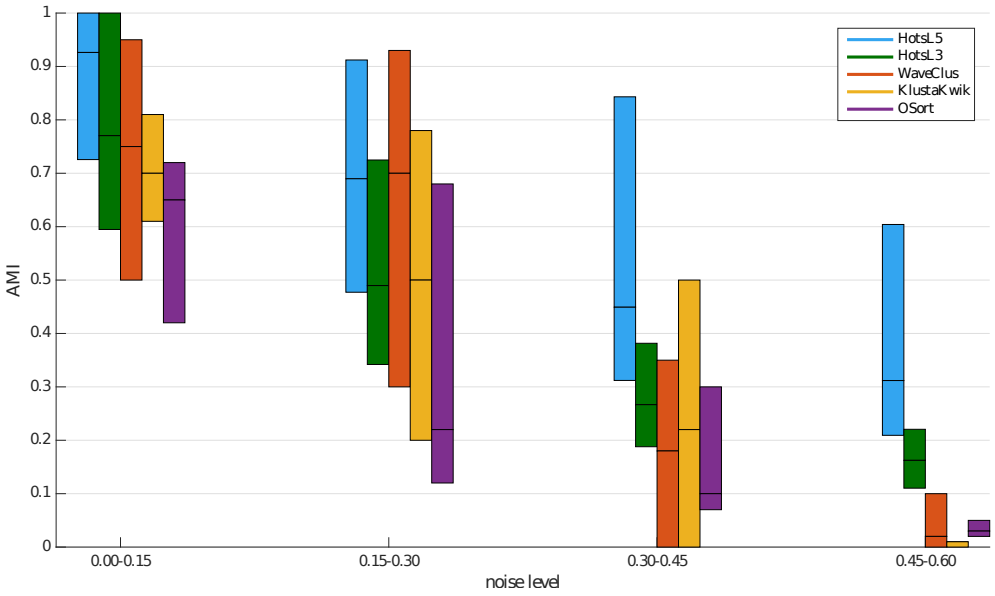


Figure 17 – Recognition rate (AMI) with median and standard deviation versus noise level and comparison to other methods. Blue bars report classification rates after layer 5, green ones after layer 3. Our method (blue and green bars) performs better than others : standard deviation is lower (median is represented in black middle line). For low noise conditions, our median is higher. For noise level between $0.15 - 0.3$, we are a bit under WaveClus, but over all the other ones. We can also see that in noisy conditions (noise level > 0.45) our algorithm is significantly better than all the others. 3 Layers run real-time.

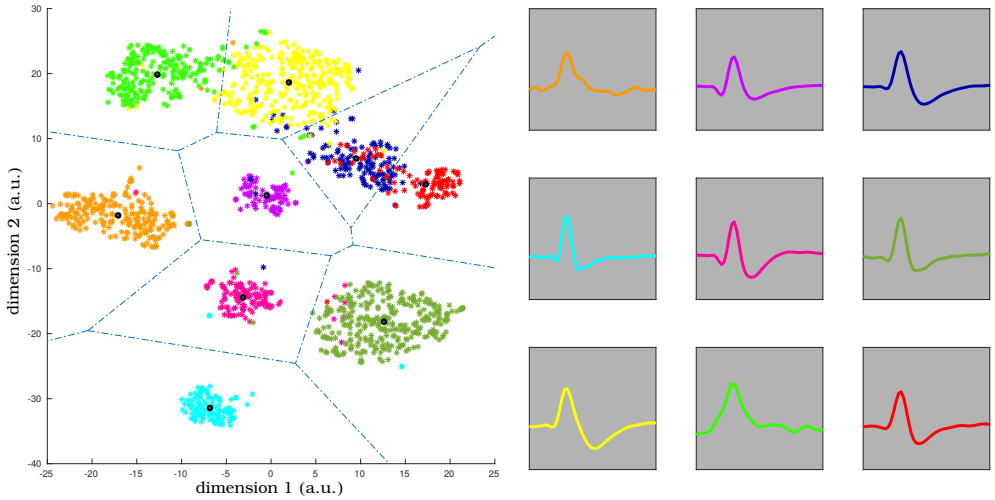


Figure 18 – Classification for the benchmarking data. (a) Voronoi diagram for the projected signatures in two dimensions and corresponding ground truth in color. Mostly, the signatures contain enough information to ensure good discriminant classification. (b) The associated temporal shapes. We can notice that our algorithm extracts signatures that are distinguishable one from each other all thereby allows good classification.

examples in two dimensions shows the ability of the algorithm to distinguish different types of spikes (Figure 18-b). Figure 17 presents the AMI score for

the presented algorithm compared to state-of-the art methods. It has to be taken in account that the only previous method working in real time is the OSort algorithm. The HotsL3 and HotsL5 presented here are for the output of the third and fifth layer of the hierarchical model. The classification score after layer 5 overcomes all the other methods. After layer 3, only for intermediate noise level ($0.15 - 0.30$) are we slightly below Waveclus. For noisy dataset (0.45 and more), the proposed method overcomes drastically all the others. This shows the very good immunity of the proposed method toward noise. Regarding the required time to process all these data, our algorithm performs real time for 1-3 layers on a standard desktop computer (Intel Core i7-4790 CPU @ 3.60GHz, 16GB of RAM, running Debian 8.5 Jessie). Figure 20 represents the computation time for 1-5 layers.

Extension to in-vivo data

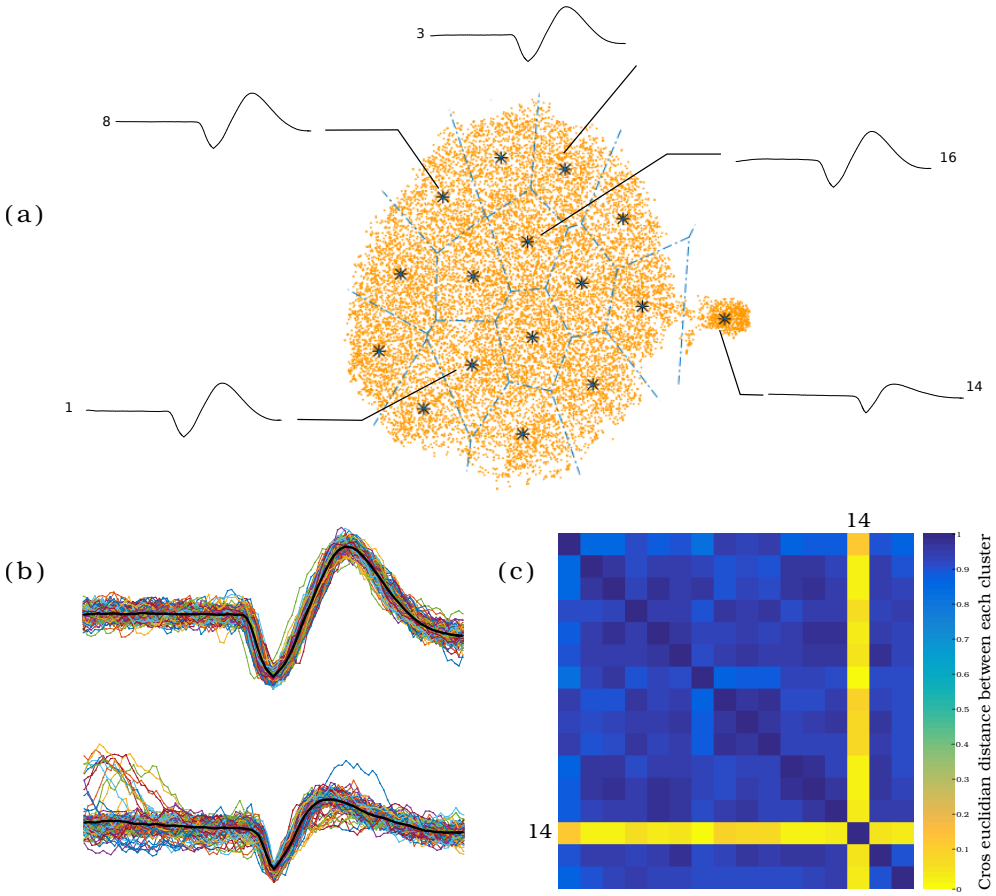


Figure 19 – Results with data from ex-vivo rat retina. (a) TSNE projection with output of the last layer. It is possible to distinguish two major types of cells (c) Clustering is able to find several possible new cells, which can be merged together. (c) The Euclidian distance allows us to merge all the clusters to estimate the best number of classes (the darker the closer). Here, we have two different classes. (b) The two found classes with mean shape (dark) and some examples (color).

Good results on semi-artificial dataset lead us to benchmark the proposed method on real data. Buzaki's lab developed new techniques for simultaneous recordings of in-vivo (patch clamp) and ex-vivo (multi electrode array)

data [50, 46]. Data from the CA1 hippocampal region of anesthetizided rats [49], containing both intra and extracellular recordings, was used. The extracellular tetrodes [107] are made of four 13-um polyimide-coated nichrome wires. The intracellular glass micropipette filed mainly with potassium solution. The extracellular data contains responses from two different cells (type A and type B), whereas the in-vivo data target only one (type A). So, the intracellular data can be used as a ground truth to validate the sorting on extracellular data. Without fine tuning of the parameters, the use of the same parameters as in previous section leads up to 90% recognition rate.

The dataset was split into 10 subsets, randomly picked, with around 30 % of type 1 cell. One subset is used for learning the clusters, and the 9 others to test the algorithm. The experience was run for 100 times. Table Table 3 shows the results of these runs. Our algorithm outperforms state-of-the art real-time spike, and is slightly lower than the best method reported on this dataset, which is not real-time. Computation time (on a Intel Core i7-4790 CPU @ 3.60GHz, 16GB of RAM, running Debian 8.5 Jessie) is about 25 seconds for one set of 80 seconds of data, which is more than 3 times faster than the real-time. Figure 20 shows the computation time versus the number of layer. We can see that the proposed method is able to handle in real-time a 3-layer architecture.

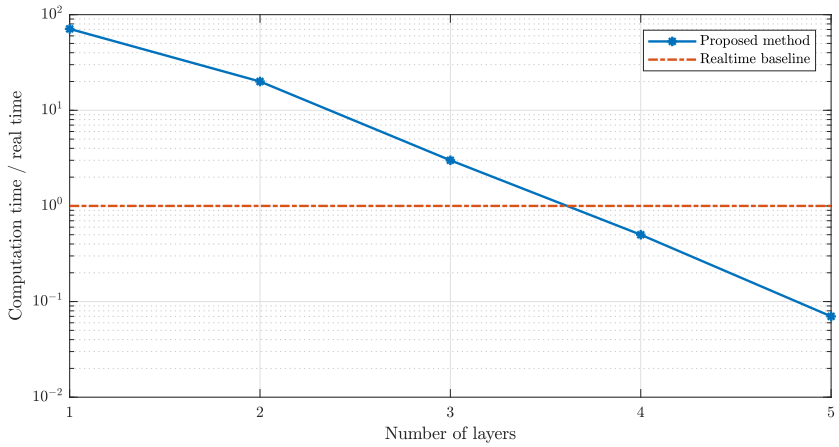


Figure 20 – Average computation time for the proposed architecture. Y-axis represent the ratio of the computation time required and the duration of the recording. Baseline of 1 (red line) is real-time. Above is faster than real-time. The proposed method is able to handle in real time a 3-layer hierarchy.

4.4 CONCLUSION

We showed that a paradigm shift in the data representation and a fully event-based hierarchical processing pipeline can lead to very good recognition rates. As stated in [59], the hereby used "event-based spatiotemporal context" is speed dependent. This may be a problem for pattern recognition, where the task is to recognize a given shape, presented in various poses and motion, but helps us for cellular recordings, as each neuron fires with a spe-

Table 3 – Recognition rate on in-vivo data. The bolted algorithms are the only one working in real-time. Our algorithm performs better than state of the art online spike-sorting, and are slightly below the known best algorithm reported on this dataset, which is not real-time.

Method	minimum	maximum	average
Shahid et. Al [115]	N/A	N/A	66%
Werner et. Al [134]	N/A	N/A	82.1%
Gasthaus et. Al [38]	N/A	N/A	Fscore 0.91%
Haessig et. Al (proposed method)	84.75%	90.22%	87.33% - Fscore 0.89%

cific dynamic that we want to extract and distinguish. Without optimisations, the proposed method is able to handle up to a 3-layer architecture, defeating state-of-the-art algorithms on publicly available datasets. This work is a major step towards efficient spike sorting, which is today a real challenge for research purpose, closed-loop applications or brain-machine interfaces [138].

4.5 DISCUSSION

In the framework of this article, the input signal is first sampled and then converted into events. Intrinsically, this approach cannot reflect the signal's exact dynamic. The linear interpolation was a first step toward time precision, but we are still limited by the sampling frequency. This work was achieved during my first year of PhD. Meanwhile, Corradi et Al. [25] developed an event-based Multi Electrode Array (MEA), which directly converts the spike voltage in events. The BioAmp chip was developed at INI (Institute of Neuroinformatics, University of Zurich and ETH Zurich). As the only available part, at the beginning of the project, was the die, I had to design, in collaboration with Federico Corradi (INI) a PCB to bond the die on, and an interface board to connect the sensor to the FPGA for readout and control operations. The bonded die is showed in Figure 21.

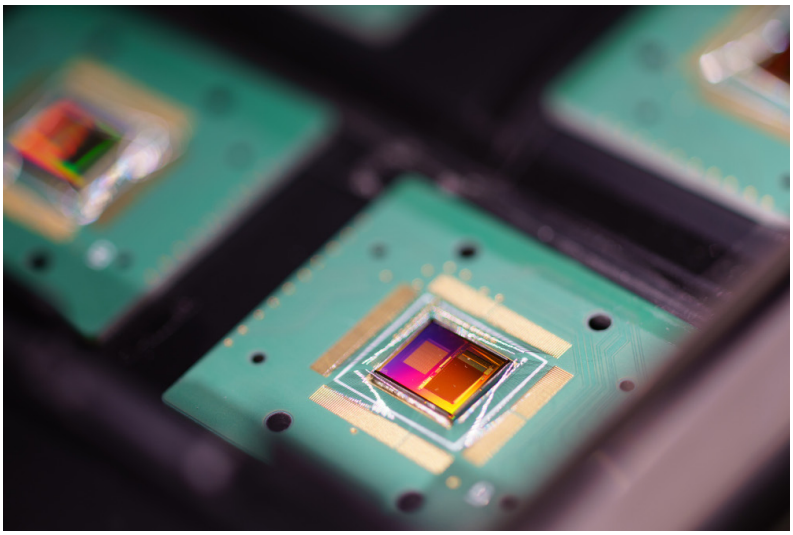


Figure 21 – One of the Bioamp chip, bonded on its PCB.

5 OPTICAL FLOW ON TRUENORTH

This chapter describes a fully spike-based neural network for optical flow estimation from Dynamic Vision Sensor data. A low power embedded implementation of the method which combines the Asynchronous Time-based Image Sensor with IBM's TrueNorth Neurosynaptic System is presented. The sensor generates spikes with sub-millisecond resolution in response to scene illumination changes. These spike are processed by a spiking neural network running on TrueNorth with a 1 millisecond resolution to accurately determine the order and time difference of spikes from neighboring pixels, and therefore infer the velocity. The spiking neural network is a variant of the Barlow Levick method for optical flow estimation. The system is evaluated on two recordings for which ground truth motion is available, and achieves an Average Endpoint Error of 11% at an estimated power budget of under 80mW for the sensor and computation.

5.1 INTRODUCTION

Estimation of optical flow is a computationally intensive task. Modern artificial systems achieve ever higher accuracy on the task at the expense of more complex algorithms and more powerful computing hardware such as GPUs. On the other hand, many biological systems are able to estimate optical flow both effectively and power-efficiently. Examples include insects such as bees and drosophilia, which rely heavily on optical flow for navigation [54], but must estimate optical flow under severe size, weight, and power constraints. While the optical flow estimated by such insect may not rival modern computer vision approaches in terms of accuracy, the estimated flow is good enough for the insects to use in navigation, and impressive considering the tight constraints under which the estimates were generated.

In artificial systems, early optical flow estimation approaches relied on methods such as velocity-tuned filters in the frequency domain [133, 47], phase-based methods [10], gradient based methods, and correlation based methods [118]. The accuracy of these methods for optical flow estimation on sequences of images has long since been surpassed by even more computationally intensive methods which estimate flow at multiple scales [2], and more recently by convolutional neural networks [31], which are mostly running on power inneficient GPUs or dedicated ASICs, which are less accurate, more general, but are more power efficient (PX4Flow [51], optical mouse). Direction selective neurons are found as early as the retina in frogs [62], cats [8], rabbits, primates, and many other animals.

A key difference between optical flow estimation in biological and artificial systems lies in the format of the data from which optical flow is extracted. Both approaches start off with light from a scene focused on an image sensor,

but most artificial systems convert this light signal into a sequence of static digital images from which optical flow must later be estimated. The retina works on a different principle, transducing the light signal into a continuous electrical signal, and later into discrete spikes for communication to the lateral geniculate nucleus and downstream visual cortex.

Not all artificial methods rely on static images though. Tanner and Mead [121] introduced the first analog two dimensional optical flow chip, extracting a global motion from a given scene on dedicated hardware. Delbruck [29] also implemented such a network in VLSI. More recently, with the new event-based cameras, some spike based techniques have been proposed [11], but they require a dedicated computer for processing. An extensive review of motion estimation designs can be found in [92].

Over the last decade, so-called silicon-retinae have matured to a level where they are now publicly available (described later in Section 5.2.2). These bio-inspired devices provide a spiking output more similar to the biological retina. Recently many papers have proposed different methods for processing data, including for estimating optical flow. Benosman *et al.* [11] proposed a plane fitting approach which estimates the motion of sharp edges. Later, Barranco *et al.* [9] proposed a method which also estimates flow at edges, but their method estimates the magnitude of the spatial and temporal image gradients at the edge and then relies on a gradient-based method for optical flow estimation. Bardow *et al.* [7] apply a variational method which simultaneously estimates both the image grayscale values and optical flow from events. By enforcing spatial and temporal smoothness constraints, their method generates optical flow estimates even for image regions where no gradient information is available.

Some optical flow estimation methods take bio-inspiration a step further and make use of biologically inspired computation using spiking neurons to extract optical flow. Examples include Brosch *et al.* implementing a phase based method [18], Conradt *et al.* implementing a Reichardt detector [109], and Orchard *et al.* [91] who simulated a method relying on synaptic delays.

In this chapter we propose a Spiking Neural Network variant of the Barlow & Levick model [8], using a silicon retina coupled with IBM's TrueNorth Neurosynaptic System.

The method exploits precise spike timing provided by the silicon retina to reliably extract motion direction and amplitude from a scene in real-time. Such methods promise to allow for low power visual motion estimation in real-time. TrueNorth is estimated to consume 70mW and the silicon retina consumes approximately 10mW (chip only, omitting FPGA for communication, which can be removed for dedicated applications). This setup can then easily be used in embedded devices such as drones or autonomous driving. Section 5.2 introduces both the model, IBM's TrueNorth Neurosynaptic System, and the silicon retina. Details of the model implementation are given in Section 5.3. Section 5.4 describes how the model was tested, and results of testing are described in Section 5.5, followed by discussion in Section 5.6.

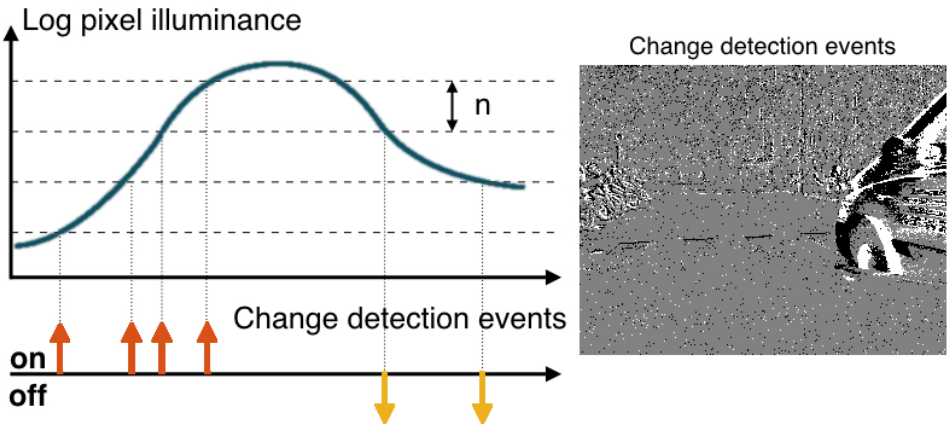


Figure 22 – Working principle of the ATIS camera. (left) A significant relative change of the luminance (programmable threshold n) generates an ON (respectively OFF) event when there is an increase (resp. decrease) in luminosity. (right) Snapshot of events over 100 ms for a car passing in front of the camera. White dots represents ON events, black ones OFF events.

5.2 BACKGROUND

5.2.1 Direction Sensitive (DS) Unit

The basis of the network, introduced in [41], is described in Fig. 23. The original model, inspired by the neural circuitry found in the rabbit's retina by Barlow and Levick [8] is based on inhibition-based direction-sensitive units, combined in motion detectors.

Fig. 23 shows a simplified Direction Selective (DS) neuron which responds (spikes) to the detection of motion in its preferred direction (rightward). The DS neuron (gray) receives inputs from two neighboring ATIS pixels (bipolar cells). Each ATIS pixel (orange and blue) will generate an output spike when stimulated by the passing of an edge. For motion in the preferred direction (rightward), the edge will pass over the orange pixel first, and the blue pixel second.

When the edge passes over the orange pixel, a spike is generated and the DS neuron is excited. The excitatory input triggers a burst of spikes from the DS neuron. The burst continues until the edge passes over the blue pixel, at which point the blue pixel will generate an output spike which inhibits the DS neuron, causing the spike burst to end. The time-length of the spike burst encodes the time taken for the edge to pass from the orange pixel to the blue pixel, thus providing information on the edge velocity in the preferred direction.

For motion in the opposite direction (leftward) the inhibition from the blue pixel will arrive before the excitation from the orange pixel. Due to the initial inhibition, the later excitation from the orange pixel is not sufficient to drive the DS neuron membrane potential above threshold, and thus the DS neuron does not spike in response to leftward motion.

The full direction is obtained with a combination of four single DS units, as shown in Fig. 24.

In implementation, there is a possibility of receiving an isolated noise spike from the orange pixel, which would cause the DS neuron to begin bursting, and continue bursting indefinitely. To handle this case, a delayed copy of the spikes from the orange pixel are used to inhibit the DS neuron, thus limiting the maximum burst length to the length of delay used.

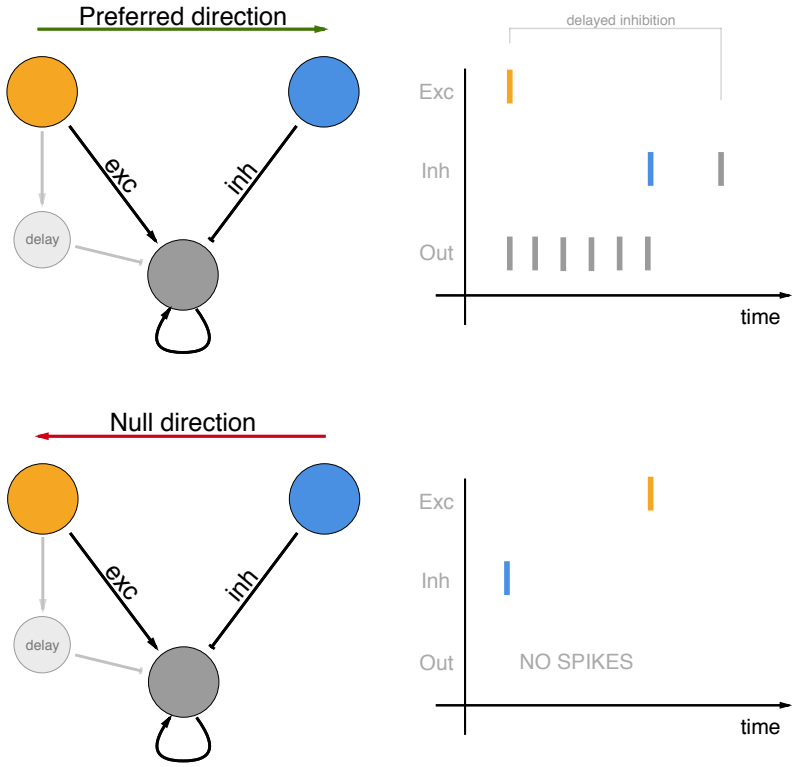


Figure 23 – Direction Sensitive (DS) unit. A stimulus from left to right first excite (yellow cell) the output (gray), which is self-excitatory until it gets an strong inhibitory input (blue cell), as a movement in the other direction (right to left) doesn't produce any output. Thus, this unit is directive to a specified direction. More, the timing between the beginning and the end of the output spiking give an information of the (inverse) velocity. The excitatory input also copies a spike to the delay unit, to inhibit the DS cell in case the natural inhibition doesn't spike. Here, the delayed inhibition doesn't have any impact. Inspired by [41].

5.2.2 Event-based Sensor

Conventional image sensors sample the visual scene at fixed temporal periods (framerate). All pixels acquire light in a synchronous fashion by integrating photons over a fixed time-period. When observing a dynamic scene, this framerate, no matter its value, will always be wrong because there is no relation whatsoever between the temporal dynamics of the scene and the acquisition period. This leads to simultaneous over-sampling and under-sampling of different parts of the scene, motion blur for moving objects, and data redundancy for static background.

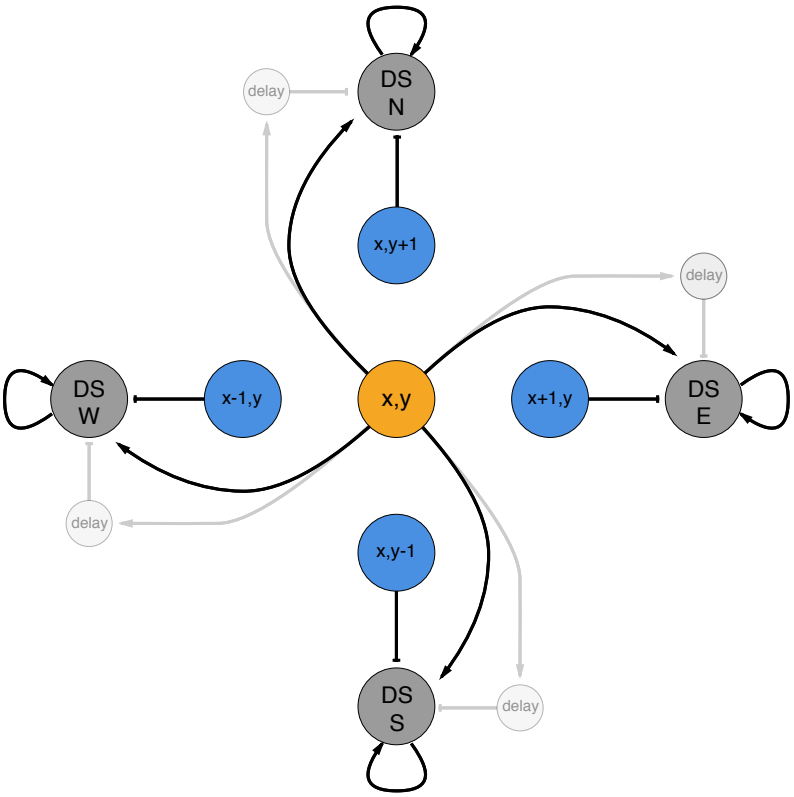


Figure 24 – Assembly of the 4 DS units (North, South, West, East). The direction of the movement is gathered by taking in consideration, for each pixel, a vertical and horizontal component. Inspired by [41].

Event-based sensors [102], also known as silicon retinae, are an alternative to fixed-frequency imaging devices. In these devices, a time-varying signal is sampled on its amplitude-axis instead of time-axis, leading to a non-uniform sampling rate that matches the temporal dynamics of the signal, as shown on Fig. 22. The Asynchronous Time-based Image Sensor (ATIS[100]) is an QVGA array of asynchronous independent pixels, each combining a change detection unit and an absolute gray level measurement unit. Each time a pixel detects a significant change in luminance in its field of view, an event is generated. Each event can be represented as a triplet $ev_i = (\mathbf{x}_i, t_i, p_i)$, ev_i being the $i-th$ event, where $\mathbf{x}_i$ is the event spatial coordinates, t_i its timestamp and p_i its polarity, indicating if the change is an increase or decrease in luminance. In this work, only the change detection unit is used.

5.2.3 The TrueNorth Environment

A TrueNorth Chip [82] consists of a network of 64×64=4096 neurosynaptic cores with programmable connectivity, synapses, and neuron parameters. Connectivity between neurons follows a block-wise scheme. Each core has 256 input axons, with programmable connectivity to any of the 256 neurons in that core. Each neuron's output can be routed to a single axon anywhere on the chip. All communications to, from, and within chip are performed asynchronously [83].

Each TrueNorth neurosynaptic core is made of 256 axons, 256×256 synapse crossbar and 256 neurons (Fig. 25 A). In this chapter, the NS1e board was used, containing 4096 cores, $1M^+$ neurons, $268M^+$ synapses, embedded in a board with a Zynq FPGA containing an ARM-core (Fig. 25 B).

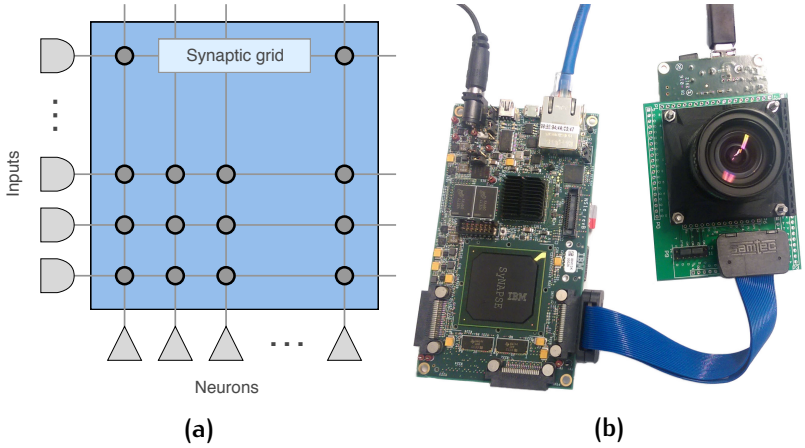


Figure 25 – a) The TrueNorth topology. Each neurosynaptic core has 256 axons, 256×256 synapse crossbar and 256 neurons. Information flows from axons to neurons gated by binary synapses, where each axon fans out, in parallel, to all neurons thus achieving a 256-fold reduction in communication volume compared to a point-to-point approach. Network operation is governed by a discrete time step. In a time step, if the synapse value for a particular axon-neuron pair is non-zero and the axon is active, then the neuron updates its state by the synaptic weight corresponding to the axon type. Next, each neuron applies a leak, and any neuron whose state exceeds its threshold fires a spike [19]. b) IBM's NS1e board (4096 cores, 1 Million neurons and 256 Million synapses) and ATIS sensor with native link used for this chapter.

5.3 IMPLEMENTATION

TrueNorth implementation of the model was achieved by carefully configuring a single tile-able TrueNorth core, thanks to the tools presented in [3], to handle a small local region of the input space, and then tiling enough copies of these cores to cover the full input image.

The TrueNorth environment neuron's behavior is as following[19] :

Synaptic integration

$$V_t = V_{t-1} + \sum_{i=0}^{N} \omega_i x_i(t)$$

Leak

$$V_t = V_t - l$$

Threshold

if $V_t > \Theta$

spike

$V_t = V_r$

where V_t is the value of the membrane potential at timestep t, ω_i the synaptic weight for the $i-th$ synapse, l the leak, Θ the threshold, V_r the reset potential, $x_i(t)$ the spike input to the synapse at time t and N the number of synapse for the considered neuron. The full neuron equations and parameters can be found in [19], figure 4. The leak is configured to, unless otherwise stated, leak towards zero. This behavior is detailed in [19].

Within each core, neurons are divided into three main modules, shown in different colors in Fig. 26. The first module (red) receives input spikes from ATIS and generates a copy of these spikes which can be used for further processing by the other two modules. The second module (blue) implements the delay required to generate the delayed inhibition for the DS neurons. The third module (green) implements the DS neurons. Each module is described further below.

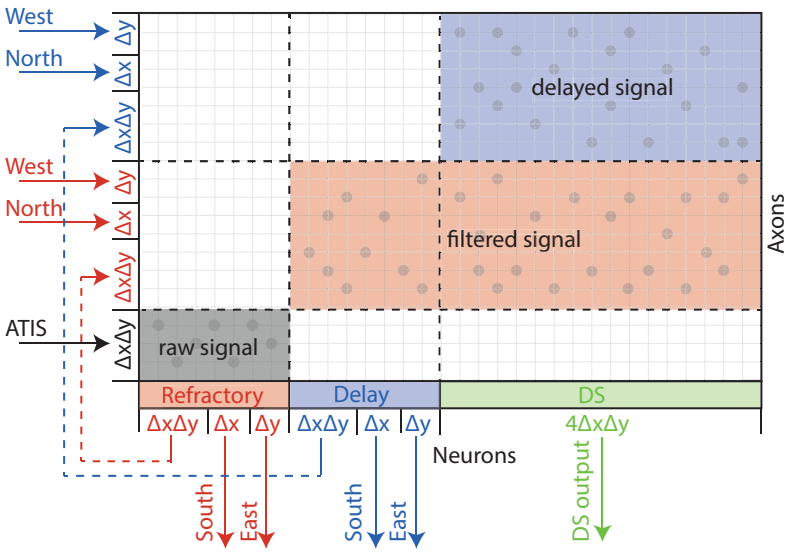


Figure 26 – Arrangement of a TrueNorth core implementing the motion model. Refractory (red), Delay (Blue), and DS (green) neurons are shown horizontally along the bottom of the core. Input axons are shown vertically on the left of the core. Text labels indicate the direction inputs are arriving from and outputs are routed to. The number of neurons (horizontal axis) and axons (vertical axis) are explicitly indicated for each part of the model. Shaded regions indicate areas where connections (synapses) exist between axons on the left and neurons at the bottom.

5.3.1 Input Module

The input stage provides a copy of the input spikes which serve as inputs to the DS neuron module and delay module. The input stage is also used to enforce a refractory period which limits the minimum allowable time between two input spikes from the same pixel by blocking the second spike if it falls within τ_r of the first.

A refractory period of τ_r (ms) is implemented using a *refractory* neuron model (see Table 4). When at rest state (membrane potential of zero), the

neuron should fire an output spike as soon as an input spike is received. This is achieved by enforcing

$$w_e + l \geqslant \Theta \tag{25}$$

where w_e is the excitatory synaptic weight.

After spiking, the neuron resets to a large negative voltage and slowly leaks back to zero, thus implementing a refractory period. The desired refractory period is achieved by setting the reset and leak values such that

$$\begin{aligned} \tau_r &\approx \tfrac{V_r}{l} \\ V_r &= -(2^n - 1), \quad n \in [\![0, 15]\!] \end{aligned} \tag{26}$$

where τ_r is the desired refractory period, V_r is the reset potential, and n the quantization of the desired value, which can only take on limited values due to constraints of the chip (here 15 bits).

Input spikes received while the neuron potential is leaking towards zero will not cause an output spike, but they will shorten the refractory period. The actual refractory period achieved, τ'_r, is given by:

$$\begin{aligned} \tau'_r &= \tfrac{V_r + w_e s}{l} \\ &= \tau_r + \tfrac{w_e s}{l} \end{aligned} \tag{27}$$

where s is the number of spikes received during the refractory period τ'_r. To ensure that the achieved refractory period τ'_r is as close as possible to the desired refractory period, τ_r, the term $w_e s/l$ must be minimized. The minimum value which still satisfies equation 25 occurs when $l = 254$, $w_e = 255$.

Connectivity on TrueNorth is bound by certain constraints, one of which is that a spike can only be routed to one core. This constraint can be overcome by implementing multiple identical neurons on the same core. Each neuron will then generate an identical output, thereby providing multiple copies, each of which can be routed to different cores.

In the case of our TrueNorth model, there are four co-located DS neurons at every pixel location, and they each require input from one of the four neighboring pixels. Thus, the spikes from some pixels serve as input to neighboring DS neurons which reside on different cores. The input stage takes care of generating copies of the spikes from these pixels and routing them to the neighboring cores.

Each core only sends spikes to its neighboring cores lying to the South and East, and each core receives input spikes from its neighboring cores in the North and West.

The number of neurons required by the input module is therefore

$$N_{in}(\Delta x, \Delta y) = \Delta x \Delta y + \Delta x + \Delta y, \tag{28}$$

where Δx and Δy specify the pixel size of the image region in x and y direction processed by a single core.

The input module implementing refraction is shown in red in Fig. 26.

5.3.2 Delay Module

The delay module implements the delay required by the delayed DS neuron inhibitory input. Delays of τ_d milliseconds (here, $\tau_d = 50$ms) are implemented using the *Delay* neuron model of Table 4.

When at rest (zero membrane potential), the neuron will remain at rest until an input spike is received. An input spike will push the membrane potential above zero, and once above zero the neuron membrane potential will leak towards a positive threshold. The delay is implemented by the time taken to leak to the positive threshold. Once the threshold is reached, an output spike is emitted, and the neuron will be reset back to the rest state.

A disadvantage of this approach is that if two input spikes arrive from the same pixel at times t_1 and t_2 such that $t_2 - t_1 < \tau_d$, only a single delayed output spike will be generated, and the output spike will be generated with delay $\tau_d - 1$ milliseconds after t_1. To avoid such complications, we use the refractory period from the input stage to limit the probability of having two spikes occur close together. More specifically, we choose τ_r such that $\tau_r > \tau_d$.

All input spikes from the region covered by the core must be delayed, and copies of the delayed spikes at the South and East boundaries must be generated for neighboring cores. Thus the total number of neurons required for the delay module, $N_{delay}(\Delta x, \Delta y)$, is

$$\begin{aligned} N_{delay}(\Delta x, \Delta y) &= N_{in}(\Delta x, \Delta y) \\ &= \Delta x \Delta y + \Delta x + \Delta y. \end{aligned} \tag{29}$$

The delay module (blue in Fig. 26) receives inputs from the refractory input module (red), as well as input from the refractory input modules of the cores to the North and West. The delay module generates outputs which feed back to the same core, as well as output for use in processing of the cores to the South and East.

5.3.3 DS Module

The DS module holds the DS neurons. The parameters for each neuron are shown in Table 4. There are four DS neurons per image location, so a core covering an image region of size Δx by Δy pixels will require

$$N_{DS}(\Delta x, \Delta y) = 4\Delta x \Delta y \tag{30}$$

neurons.

The output neuron of the DS module should be self-excitatory until it receives a strong inhibitory spike, as shown in Figure 23. This behavior was achieved by setting the reset potential V_r to be higher than the threshold Θ. Then, when the membrane potential crosses the threshold, the cell enters a self-excitatory state until a strong inhibition event is received.

Sometimes, due to noise, the inhibitory spike can be missing. To prevent a cell to spike indefinitely, and second strong inhibition spike is generated and delayed (typically 100ms) as the excitation spike is generated. So, if the inhibition is not received, this delayed spike will reset the neuron, and thus

Table 4 – Neuron parameters

Parameter	Symbol	Refractory	DS neuron	Delay
excitatory weight (mV)	w_e	255	150	1
inhibitory weight (mV)	w_i	0	50	0
threshold (mV)	Θ	1	125	τ_d
leak (mV/ms)	l	-254	-1	1
reset potential (mV)	V_r	$\tau_r.l$	127	0
negative floor (mV)	β	$\tau_r.l$	-50	0
Delay (ms)	τ_d	50		
Refractory period (ms)	τ_r	50		

putting it in a steady state. It has to be recalled that not all the noise events will trigger a flow computation, as a valid flow sequence consists of two spikes in the same neighborhood.

The DS neurons are shown in green in Fig. 26 and receive input from both the input refractory module, and the delay module of the current core, as well as from the cores to the North and West.

5.3.4 Parameters

TrueNorth neurons are a variant of linear leaky integrate and fire model with 23 tunable parameters, as mentioned in [19]. Table 4 contains the neuron parameters for each population.

The total number of neurons $N_{\sum}$ required by a core is

$$\begin{aligned} N_{\sum}(\Delta x, \Delta y) &= N_{in}(\Delta x, \Delta y) + N_{delay}(\Delta x, \Delta y) \\ &\quad + N_{DS}(\Delta x, \Delta y) \\ &= 6\Delta x\Delta y + 2\Delta x + 2\Delta y \\ N_{\sum}(\Delta x, \Delta y) &\leqslant 256. \end{aligned} \tag{31}$$

The values $\Delta X = 6$ and $\Delta Y = 6$ maximize the image area processed by the core, subject to the constraint that the core only has 256 neurons. The number of axons used is less than the number of neurons, so neurons are the limiting factor.

285 identity cores are used to relay the ATIS input to the motion model (see Section 5.3.6). For the motion model to cover the 304×240 pixel input image requires another $304/6 \times 240/6 \approx 51 \times 40 = 2040$ cores. In total the TrueNorth model uses $2040 + 285 = 2325$ cores to compute motion at all locations in the input image space. 240 neurons are used per core (a usage of 256 neurons per core is achievable if $\Delta X = 18$ and $\Delta Y = 2$, but each core would still only be processing 36 pixels, and thus such an arrangement is less efficient).

5.3.5 Interpreting the Result

To interpret the output spikes from the network at a particular pixel location, the outputs of all four of the DS neurons located at the pixel are combined. The same input spike will excite all four DS neurons, so they should all start bursting simultaneously (unless they are already inhibited). We use the notation t_{x^+} to denote the length of the burst from the neuron sensitive to motion in the positive x-direction. The velocity can then be calculated using

$$\begin{aligned} t_x &= (t_{x^+} - t_{x^-})/\delta_x \\ t_y &= (t_{y^+} - t_{y^-})/\delta_y \\ v_x &= t_x/(t_x^2 + t_y^2) \\ v_y &= t_y/(t_x^2 + t_y^2) \end{aligned} \tag{32}$$

where δ_x and δ_y represent the inter-pixel distance in the x and y direction respectively (in our case, $\delta_x = \delta_y = 1$). v_x and v_y indicate the component of velocity in the x and y directions respectively, in units of pixels per millisecond. If the inhibition is missing, the flow computation is discarded.

5.3.6 ATIS-TrueNorth link

The ATIS and TrueNorth chips both use variants of the Address Event-Representation (AER) protocol [17]. However, the two are not directly compatible, so the Opal Kelly XEM6010 board with Xilinx Spartan 6 FPGA which powers and configures ATIS is also used to convert the ATIS AER signals to TrueNorth AER signals.

The ATIS AER data consists of an absolute pixel address and a polarity, all of which are communicated in parallel when the AER request line is active.

The TrueNorth AER data consists of a relative core address, an axon address, and a 4 bit target time. The data is communicated in two phases, with half the bits being communicated in parallel during each phase.

The TrueNorth reset and 1kHz clock tick signals are shared with the Opal Kelly board so that it can keep track of the state of TrueNorth's internal 4 bit time. All events are communicated to TrueNorth with a target time 2ms later than the current time.

A one-to-one mapping from ATIS pixels to TrueNorth neurons is generated such that each TrueNorth core accepts events from a 16×16 pixel region, with polarity being ignored (all events are fed in the system, regardless their polarity). An array of 19×15 = 285 identity cores is instantiated at the physical core locations targeted by the ATIS interface. These identity cores generate a copy of the spikes they receive, which can then be routed to the rest of the TrueNorth model.

This interface arrangement uses an extra 285 cores, but allows the model to be rapidly changed by only reconfiguring TrueNorth, instead of having to reprogram the ATIS FPGA to target different cores and neurons for different TrueNorth models.

5.4 TESTING

5.4.1 Sources of Visual Data

Two sources of visual data are used in this work: a recording of a black pipe rotating in front of a white background, originally presented in [91], and a recording of a rotating spiral [92]. The pipe and spiral sequences are both ATIS recordings in which ground truth for motion can be predicted to quantify the accuracy of the output. The pipe and spiral recordings are shown in Fig. 27. This motion is modeled as described below.

5.4.2 Modeling Motion for the Rotating Pipe

We use a recording of just over half rotation of the pipe due to symmetry (the second half of the rotation will look almost identical to the first). A full rotation takes roughly 2.85 seconds, so we use a 1.5 second recording. The rotating pipe was modeled as having two parallel edges spaced a width of w apart from each other and rotating about a point centered between them. Motion of the pipe could then be modeled using

$$\begin{aligned}
R(t) &= \begin{bmatrix} \cos(t\omega_t) & -\sin(t\omega_t) \\ \sin(t\omega_t) & \cos(t\omega_t) \end{bmatrix} \\
\text{Location}(l,t) &= \begin{bmatrix} x_c \\ y_c \end{bmatrix} + R(t) \begin{bmatrix} \pm\frac{w}{2} \\ l \end{bmatrix} \\
\text{Direction}(t,l) &= \begin{cases} t\omega_t & l \leqslant 0 \\ t\omega_t + \pi & l > 0 \end{cases} \\
\text{Speed}(l) &= |l|\omega_t \\
l &\in [-150, 150] \\
t &\in [0, \tfrac{\pi}{|\omega_t|}] \\
\omega_t &= 2.21 rad/s
\end{aligned} \tag{33}$$

where $R(t)$ is a rotation matrix which varies with time (t) to model rotation of the pipe, ω_t is the angular velocity of the pipe, $\text{Location}(l,t)$ gives the pixel location, $[x,y]^T$, of points on the pipe as a function of time, their position on the edge of the pipe l, and the location of the center of the pipe $[x_c, y_c]^T$. The speed of each point is dependent only on its position on the pipe l, while the direction depends on t and l because one half of the pipe is moving in the opposite direction to the other. The pipe has a finite length of 300 pixels, thereby limiting l. The recording is long enough for half a rotation of the pipe, thereby limiting t. The speed and direction given by (33) are the components perpendicular to the edges of the pipe. The direction of the x and y axes are shown top left of Fig. 27.

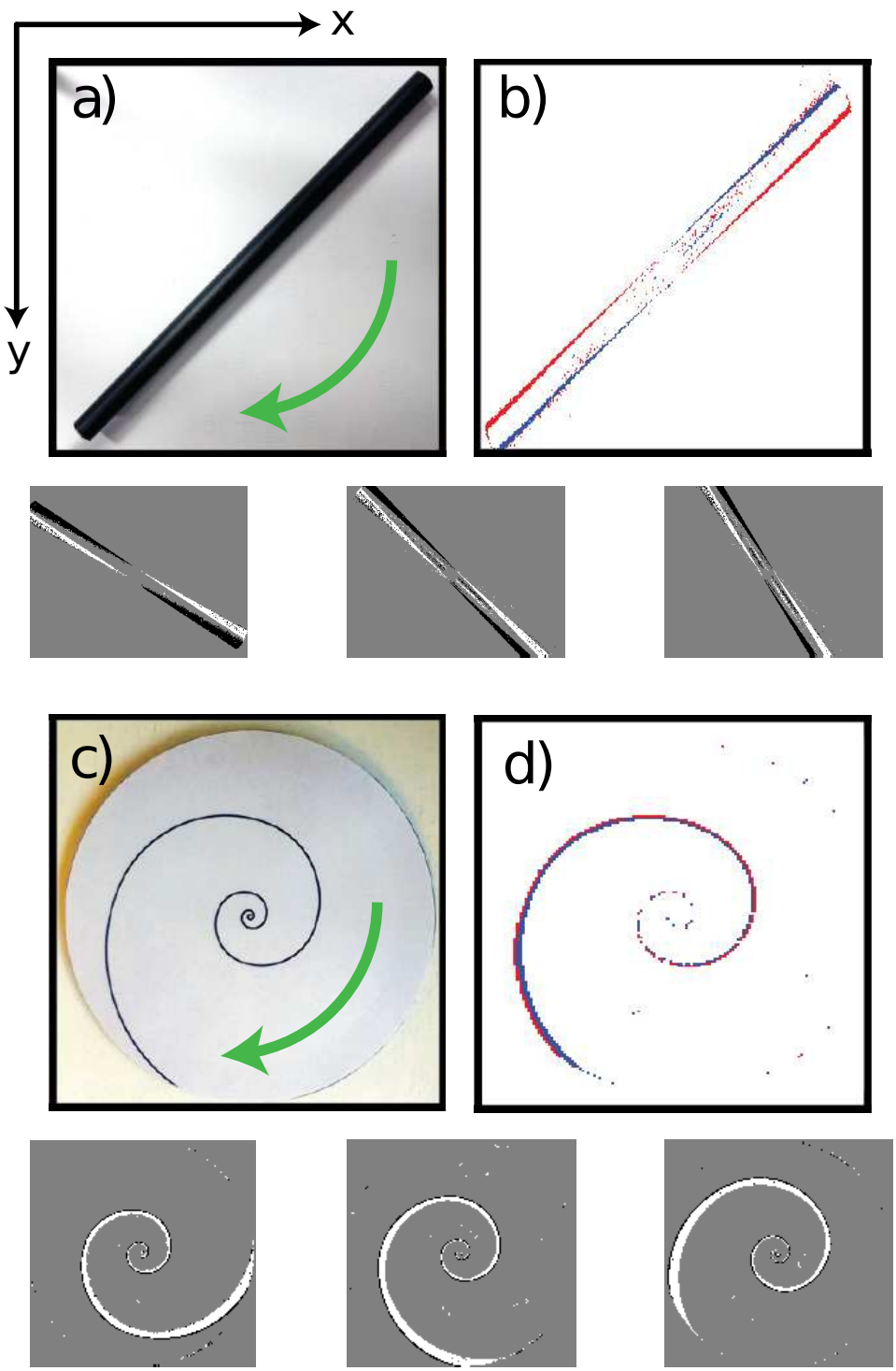


Figure 27 – The two recordings used in this chapter. (a) and (b) show images of a rotating pipe and spiral respectively from the sensor's viewpoint. The direction of rotation is shown by the green arrows inset. (c) and (d) show 10ms each of the pipe and spiral recordings. Red points indicate OFF-events (decreases in intensity) while blue points indicate ON-events (increases in intensity). Axes directions used in (33) and (34) are shown in the top left.

5.4.3 Modeling Motion for the Rotating Spiral

The spiral stimulus does not exhibit the same symmetry as the pipe, so we use a recording of a full rotation of the spiral, which takes 0.5 seconds.

The shape of the spiral stimulus was parameterized by angle with a variable θ_0 from which the speed, direction, and location of motion can be modeled using

$$\begin{aligned}
r(\theta_0) &= 2^{\frac{\theta_0}{\pi}} \\
\text{Location}(\theta_0, t) &= \begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} r(\theta_0)\cos(-\theta_0 + t\omega_t) \\ r(\theta_0)\sin(-\theta_0 + t\omega_t) \end{bmatrix} \\
\text{Speed}(\theta_0, t) &= 2^{\frac{\theta_0}{\pi}} \frac{\ln 2}{\pi} \omega_t \frac{1}{\cos\left(\frac{\ln 2}{\pi\omega_t}\right)} \\
\text{Direction}(\theta_0, t) &= -\theta_0 + t\omega_t + \sin\left(\frac{\ln 2}{\pi\omega_t}\right) \\
\theta_0 &\in [0, 20] \\
t &\in [0, \tfrac{\pi}{|\omega_t|}|] \\
\omega_t &= -12.57 \text{rad/s}
\end{aligned} \tag{34}$$

where $r(\theta_0)$ is the radial distance to the spiral from the center, $[x_0, y_0]^T$. $\text{Location}(\theta_0, t)$ describes the $[x, y]^T$ location of the spiral edges. $\text{Speed}(\theta_0, t)$ and $\text{Direction}(\theta_0, t)$ give the speed and direction of motion respectively. The Cosine and Sine terms in the speed and direction equations account for the fact that the spiral is not perpendicular to the radial vector (second term in $\text{Location}(\theta_0, t)$).

5.4.4 Error Metrics

We use three main metrics to evaluate the results. Average Angular Error (AAE), Average Endpoint Error (AEE), and density estimation.

The first two error metrics (AAE and AEE) rely on ground truth motion being available from (33) and (34). Sometimes sensor noise spikes result in flow being estimated where there is no image motion. The true flow is assumed to be zero for all locations which do not lie on an edge of the bar or spiral. For such locations, the error is equal to the flow estimate itself.

AAE is simply the average absolute error in angle between the computed flow $\vec{V} = (v_1, v_2)$ and the ground truth $\vec{Gt} = (gt_1, gt_2)$.

$$AAE = (\vec{V}, \vec{Gt}) = atan\left(\frac{gt_2 - v2}{gt_1 - v_1}\right) \tag{35}$$

AEE (equation 23 from [5]) takes into account errors in both speed and direction. It is defined as the squared root of sum of the the squared errors between the estimated vector and ground truth.

$$AEE = \sqrt{\frac{(v_1 - gt_1)^2 + (v_2 - gt_2)^2}{gt_1^2 + gt_2^2}} \tag{36}$$

Density (d) provides a measure of the ratio of optical flow estimates generated by the model to the number of input events fed to the model. A higher

density is better so long as it is not accompanied by a significant decrease in accuracy. Many event-based methods quote this metric to avoid a situation where incredibly high accuracies are obtained by discarding all but the very best flow estimates.

$$d = \frac{N_{valid}}{N_{all}} \tag{37}$$

where N_{valid} is the number of optical flow estimates, and N_{all} is the total number of input events.

5.5 RESULTS

Output spikes are sent from the board as UDP packets. A simple UDP receiver code for visualizing the output spikes was developed that captures UDP packets, decodes them and plots the results of the flow (direction and velocity). All the error analyses were done offline.

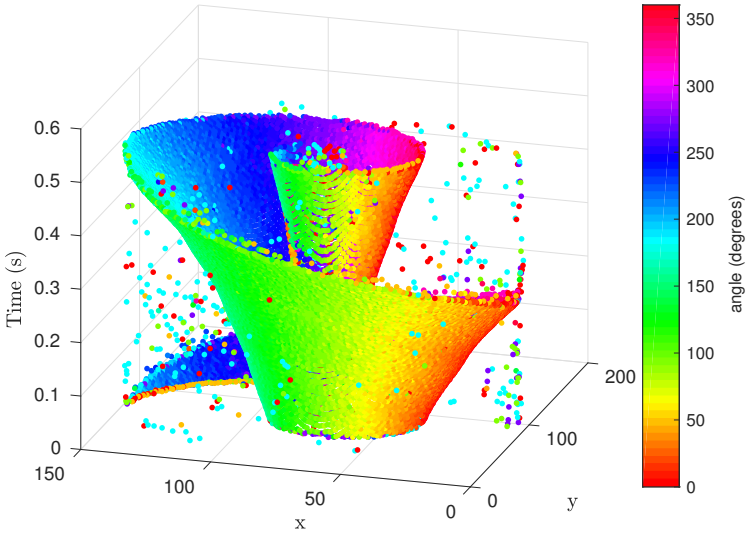


Figure 28 – 3D representation of network result on the Spiral data. For clarity purpose, only the direction is here represented. The direction of movement is color-coded in the HSV space. The individual isolated spikes are flow estimates generated in response to sensor noise. Video available online [128].

The implemented network, as presented in Section 5.3, uses as parameters $\Delta X = 6$ and $\Delta Y = 6$, leading to a need of 2325 cores, representing 558000 neurons. Fig. 27 shows snapshots of the input data (rotating bar and rotating spiral). Feeding this network with these input gives results shown in Figs. 28-31.

Fig. 28 shows a 3D representation of the optical flow estimates arising from the spiral data. Only the estimated direction of movement is shown, indicated by hue.

Fig. 29 shows the AAE for the same spiral recording, in which the direction of movement is estimate with an AAE of $\bar{\epsilon} \simeq 8.5\,\text{deg}$. This error is mainly located at the ends of the bar, where no neighborhood is available, thus giving incorrect measurements.

Fig. 30 shows the AEE for the spiral data. The previously described network is able to extract the optical flow for the spiral with an average AEE of 11.3%. A video of the spiral results is available online at [128].

Fig. 31 shows similar angular error for the pipe data. The plots appear truncated because the bar was not entirely in the field of view of the camera. As for the spiral, most of the errors are located at the edges of the pipe, also due to a lack of neighboring events. The AAE is 8.7° for this dataset. A video of the pipe results can be found online at [127].

Fig. 32 shows a snapshot of the output from data captured with the sensor hand-held in the lab. We provide this example in order to show the ability of the network to extract flow in a more complex scene, but ground truth and error estimates are not available. A video of this data is available online at [126].

The percentage of flow computations triggered by noise is 3.95%. We consider a flow computation to have been triggered by noise if the time and location of the flow estimate is not predicted by (33) or (34).

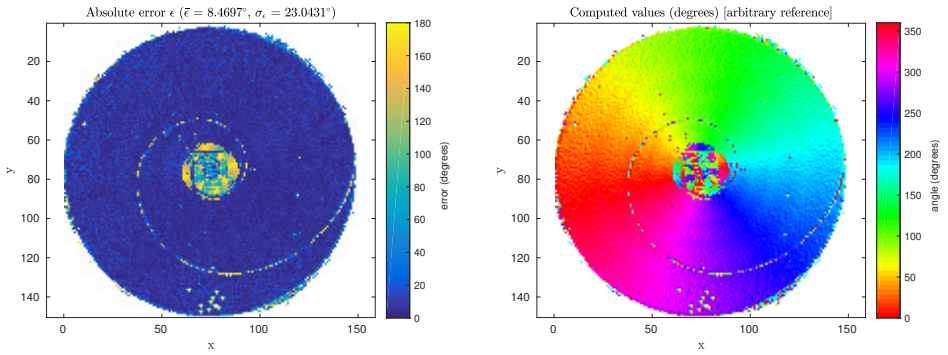


Figure 29 – Rotating spiral. a) Absolute error. One can notice that the error is mainly located at the end of edges, due to the lack of adjacent active pixels. In some extend, it is impossible to compute a flow here. b) Top view of the output of the network.

5.6 DISCUSSION

The model presented here is capable of estimating visual motion in real-time with good accuracy, but there are some limitations.

The method described uses 57% of the cores available on TrueNorth to estimate optical flow at every pixel over the full ATIS resolution of 304×240. To extend the method to operate at higher resolutions, such as the prototype 640×480 pixel DAVIS sensor, would require either using multiple TrueNorth chips, or only computing optical flow at every second pixel in the x and y directions.

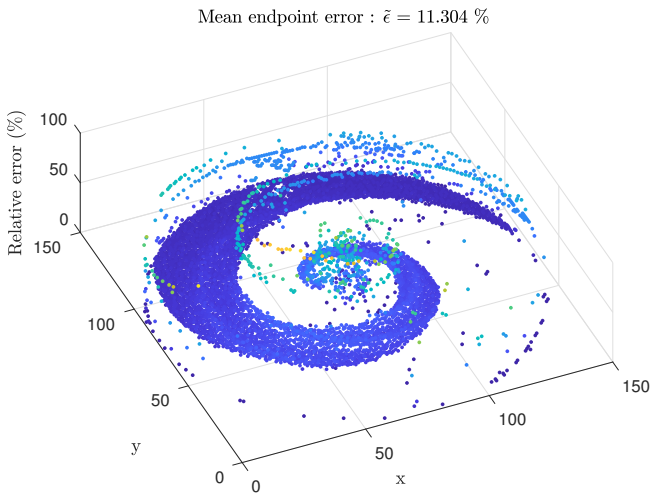
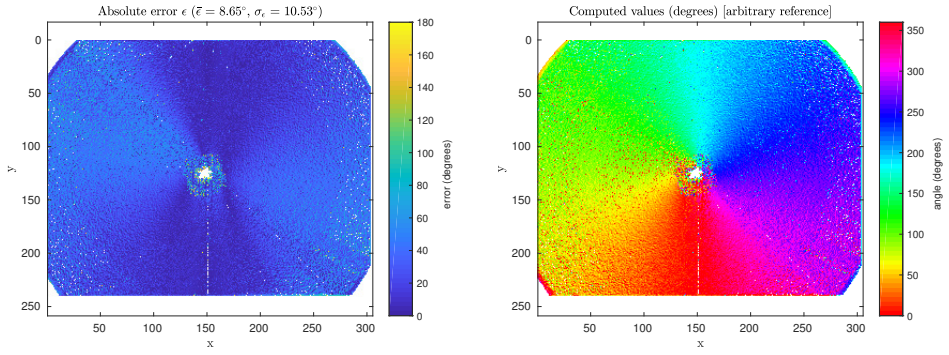


Figure 30 – Average Endpoint Error (AEE) for the spiral input. Our network is able to reconstruct the velocity vector with an average error of 11%, error which is mainly located at the edges of the spiral, where it is impossible to know the speed by our method.



(a) **(b)**

Figure 31 – Rotating pipe. a) Absolute error. One can notice that the error is mainly located at the end of edges, due to the lack of adjacent active pixels. In some extend, it is impossible to compute a flow here. b) Top view of the output of the network. Video available online [127].

Of the 256 neurons available on each TrueNorth core, 16 are unused. Of the 240 remaining neurons, 24 (10%) are used to copy spike signals across the core boundaries to prevent motion estimation artifacts at the core boundary. Since copying the signal across the core boundary introduces a 1 tick delay, the inputs to each core must also be delayed 1 tick before processing, which uses another 6×6=36 of the 240 used neurons. The block connectivity of TrueNorth has many advantages for parallelizing local computations in each core, but at the cost of requiring spikes to be copied between neighboring blocks, which ends up using $36 + 16 = 52$ neurons per core.

Another constraint is imposed by the time resolution of TrueNorth, which is operating at a 1ms tick interval. This time quantization results in each DS unit's output speeds being quantized to $1/n, n \in \mathbb{N}$ pixels/ms. Such quantization can result in large error for large speeds (since the quantized

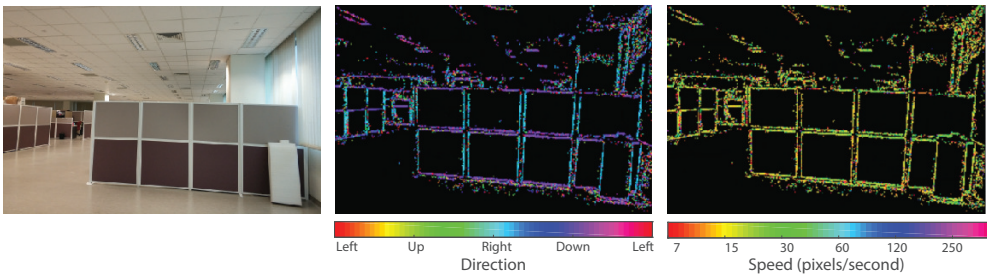


Figure 32 – Snapshot from a video showing optical flow output from TrueNorth for data captured while moving the ATIS sensor by hand in the lab. The leftmost image shows the scene from a similar angle captured using a cellphone camera. The middle shows the direction of motion detected, and the right shows the speed. Strong vertical and horizontal edges come out clearly, but the direction of motion differs because the network estimates the normal component of optical flow.

bins are further apart at high speed), and speeds faster than 1 pixel/ms will be detected as 0. The maximum detectable speed could be increased by placing DS input pixels further apart. A spacing of k pixels between the DS unit input pixels would result in a maximum possible speed of k pixels/ms. However, this approach would aggravate the edge correspondence problem, because it becomes less likely that the two pixels are seeing the same edge. Another possible method for increasing the maximum detectable speed is to reduce the tick period for TrueNorth. While this is definitely possible, keeping in mind that lowering the tick period may result in problems with spike delivery (routing issues) and may increase power consumption, a maximum speed of 1 pixel/ms was deemed good enough (objects shouldn't cross the visual field in less than 304ms *-i.e. sensor width-*. Proper lens selection may be done according to the desired application), and reducing the tick period of TrueNorth was not explored further.
The slowest speed which can be detected is $1/\tau_r$, set by the length of the delay, τ_r. Increasing the spacing of DS pixels to k would also increase this minimum detectable speed to k/τ_r. There is a trade-off between the slowest detectable stimulus, and the minimum allowable time between two subsequent stimuli. For two nearby, fast moving stimuli to each be uniquely detected and measured, their onsets must be spaced as least τ_r ms apart. Thus for fast moving stimuli it is desirable to have τ_r be small, but for slow stimuli, we require τ_r to be large.

Spike IO bandwidth of the TrueNorth chip also poses a constraint. The TrueNorth chip has four spike IO ports (North South East West), each capable of 40k spikes per millisecond. On the NS1e development platform used in this work, all outputs spikes are communicated to a Xilinx Zynq through one port. Although 40k spikes/ms is a lot, the model presented in this chapter does generate a lot of output spikes. The worst case scenario happens in complex scenes with lots of slowly moving edges. Edge density increases the number of DS units which respond, while slow moving edges generate a lot more spikes than fast moving edges. An isolated noise event at a single pixel is a particularly bad culprit for generating spikes since it will simulta-

neously activate all four 4 DS units located at that pixel, and each of these DS units will generate a train of τ_r spikes before self-inhibiting.

If too many output spikes are generated, the TrueNorth chip tick period will extend to allow time for the spikes to be communicated, which will distort the estimated flow (which assumes a 1ms tick). In the case where pre-recorded data is being run through the model (instead of live ATIS data), the TrueNorth Neurosynaptic System will input spikes in accordance with the slowed down tick, thereby still allowing the model to estimate speeds accurately, but the system will run slower than real-time. On-chip spike communication is much faster than off-chip spike communication, so in future a method which interprets velocity on-chip and more efficiently communicates the results off-chip can help to mitigate the IO constraint. The model described in this chapter can be classified as a token based method, where a token is generated whenever an edge passes over a pixel. The actual gradient of the edge is never estimated, the gradient just has to be large enough to trigger a spike from the ATIS pixel. Large gradients may result in multiple spikes in response to a single edge, but the refractory period of the first stage will eliminate these secondary spikes. Much like many other event-based optical flow methods [41, 11, 9], this method only estimates flow at edges, and it estimates normal flow (optical flow normal to the edge direction). However, some event-based methods do estimate gradients for use in optical flow computation [9], or use global constraints to estimate optical flow for image regions where little or no gradient is present [7].
The data used for this chapter, specifically the spiral one, were chosen because of their wide spectrum in terms of directions, the full range is covered within on rotation, and speeds, as the linear speeds depends on the radius for a constant rotational speed. Currently the output spike signals must be interpreted off chip to extract the velocity information. Since 43% of the TrueNorth cores are unused, there remains room to develop on-chip interpretation of the spikes to extract velocity. However, the output of the TrueNorth chip is always in spiking format, so such development work would simplify, but not eliminate, the off-chip velocity extraction.

5.7 CONCLUSION

A SNN method for normal optical flow computation from silicon retina data has been proposed. The method is capable of measuring the speed of edges in the range of 1/50 pixels/ms to 1 pixel/ms in real time. The light signal coming into the vision sensor is transduced into spikes by a silicon retina. These spikes are then fed into the SNN for processing, which provides an output estimate also encoded as spikes. This network, consuming 80m*W* (70m*W* for TrueNorth, 10m*W* for the ATIS, omitting FPGA used for communication, which can be removed for targeted applications), running real time and using 2325 cores, 558000 neurons, is able to extract optical flow with both a low Average Angular Error rate below 10 degrees, and Average Endpoint Error of 11% (for the spiral), for a density estimation of 51%.

6 NEUROMORPHIC NETWORKS ON SPINNAKER

This chapter introduces a pure, event-driven visual encoding approach that uses precise timing mechanisms to design new computation techniques in visual processing, with the use of a real-time neuromimetic platform specifically designed to efficiently exploit the eventdriver nature of the proposed algorithms. The research presented in this chapter resulted in a full event-driven visual processing system linking a neuromorphic retina[100] directly to the SpiNNaker system by an Asynchronous Event Representation (AER) bus. I implemented event-driven early vision models (optical flow, same network as the one described in Chapter 5) and 3D stereovision [73] [111] in the SpiNNaker board using a precise timing mechanism. The work presented in this chapter is part of the work package *WP11.3.5 : Asynchronous Computational Retina* of the Human Brain Project, and specifically concerns the milestones 11.3.5.1, 11.3.5.2, 11.3.5.5 and 11.3.5.7.

6.1 INTRODUCTION

Biomimetic event-driven time-based vision sensors are a novel class of vision devices that - like the biological retina - are driven by "events" happening within the visual scene. They are not like conventional vision sensors, which are driven by artificially created timing and control signals (e.g. frame clock) that have no relation whatsoever to the source of the visual information. This work aimed to introduce a whole neuromorphic processing framework implementing optical flow and stereo disparity detector. I relied only on the output of a circuit contained in the Asynchronous Time-Based Image Sensor (ATIS) pixels [100] that detects relative changes in pixel log luminance over time. As soon as a change is detected, the process of communicating this change event off-chip is initiated. Off-chip communication executes with low latency (on the order of microseconds), ensuring that the time at which a change event is readout from the ATIS inherently represents the time at which the change was detected. This asynchronous low-latency readout scheme provides the high temporal resolution change detection data our features rely on. We discuss two types of change detection events: "ON" events and "OFF" events, which respectively indicate that an increase or decrease in log pixel intensity has been detected. This representation of the visual information fits the SpiNNaker massively parallel multi-core architecture. It benefits from SpiNNaker's ability to simulate millions of neurons, and makes use of its computing power to develop and map event-driven computation architecture. The flow is inspired by the neural circuitry found in the rabbit's retina by Barlow and Levick [8], based on inhibition-based direction-sensitive units, combined in motion detectors. A single direction sensitive (DS) unit provides information only if the object

moves in its preferred direction, otherwise it stays silent. To extract a 2D time-of-travel vector we combined four DS units together in a single motion detector. A 64x64 pixels optical flow was implemented and runs real-time on the platform. We performed stereovision using the asynchronous high temporal resolution properties of the ATIS camera as developed in [111], using a modified version of a human stereopsis from Marr & Poggio [73]. The combination of spatial and temporal constraints fully uses the high temporal resolution of neuromorphic retinas. It allows us to produce an optimal stereo algorithm that is able to perform depth computation by detecting coactive pixels laying on the same epipolar lines.

6.2 THE SPINNAKER PLATFORM

« The spiking neural network architecture (SpiNNaker) project aims to deliver a massively parallel million-core computer whose interconnect architecture is inspired by the connectivity characteristics of the mammalian brain, and which is suited to the modeling of large-scale spiking neural networks in biological real time. Specifically, the interconnexion allows the transmission of a very large number of very small data packets, each conveying explicitly the source, and implicitly the time, of a single neural action potential or *spike*. » [35].

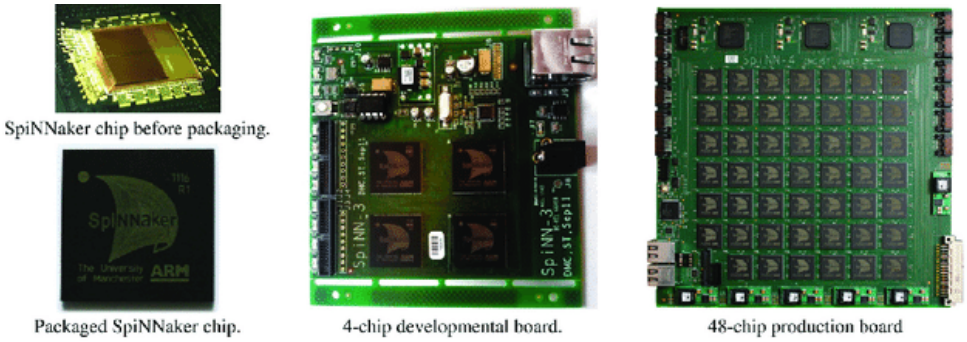


Figure 33 – SpiNNaker chips and development boards. From [87].

6.3 INTERFACING ONE EVENT-BASED CAMERA TO SPINNAKER

The ATIS camera used in this work is composed of two electronic boards. The first one, denoted the *sensor board* contains the ATIS chip itself with all its required components. The second one is an Opal Kelly XEM3010 FPGA board housing a XC3S1500-4FG320 Spartan-3 FPGA from Xilinx.

We designed an interface boards which can be plugged in between these two existing PCBs (see Fig. 36) in order to add the driving electronics and connectors required to interface the ATIS camera to the SpiNNaker system. This interface board contains the connector required for the SpiNNaker link

(see [123, 99]) and the linking logic. A view of this PCB is presented Fig. 34, showing the different components sitting on the board.

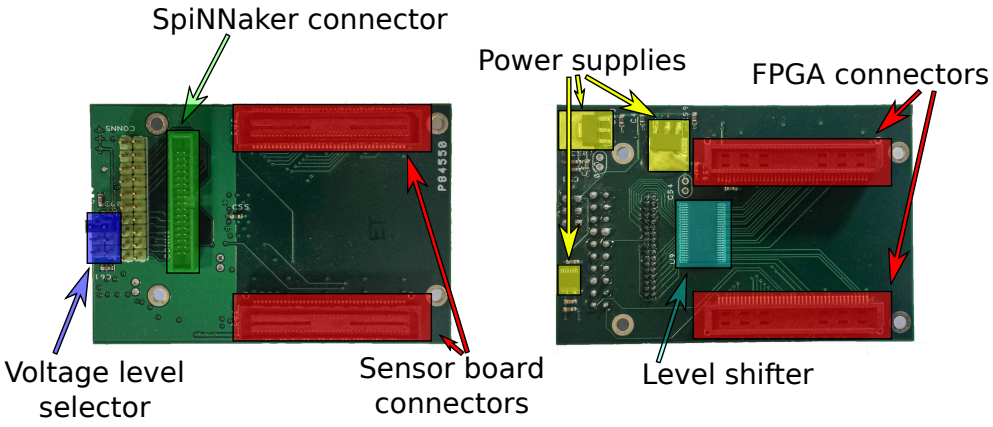


Figure 34 – Interface board used to feed data from an ATIS camera to SpiNNaker. The interface board sits in between the sensor board and the FPGA board driving the camera.

ATIS events occupy a routing space of 9 + 8 + 1 = 18 bits, as the x-coordinate can be mapped in the 0-303 range and the y-coordinate can be mapped between 0-239 and the polarity bit can assume values in the 0, 1 range accordingly to the light intensity change direction.

An original SpiNNaker event on the other side is represented by a 32 bit routing key/address (8+8 for the x,y chip coordinates, 5 bits for the core coordinate and 11 bits for the neuron id within a core). An ATIS events need therefore to be projected in the 32-bit routing space of SpiNNaker. This is done using the techniques described below, which derives from [99].

Both solutions rely on the idea to represent the spiking sensor as a virtual SpiNNaker chip, external to an original physical mesh and to convert the addresses to the same format used by SpiNNaker. By doing so the sensor can seamlessly be integrated in the network simulated on SpiNNaker; the sensor itself emits spikes which are no different than the ones produced by other SpiNNaker chips; its interconnection is therefore transparent to the system. In other words, spiking sensors are assigned fictional, virtual chips, which are not physically present on a SpiNNaker board. Sensors directly feed their spiking data into the SpiNNaker interconnect through the bespoken SpiNNaker link or through the SATA interface.

The mapping mechanism and the SpiNNaker packet structure are shown in Figure 35 (after [99]).

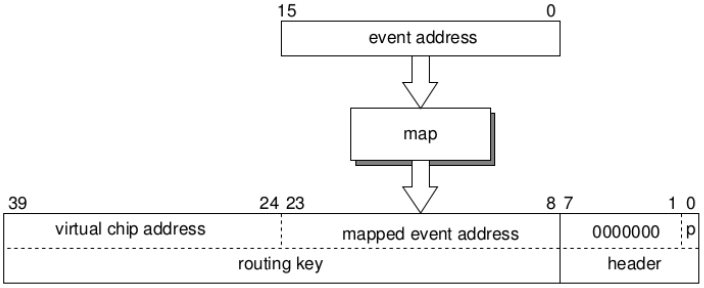
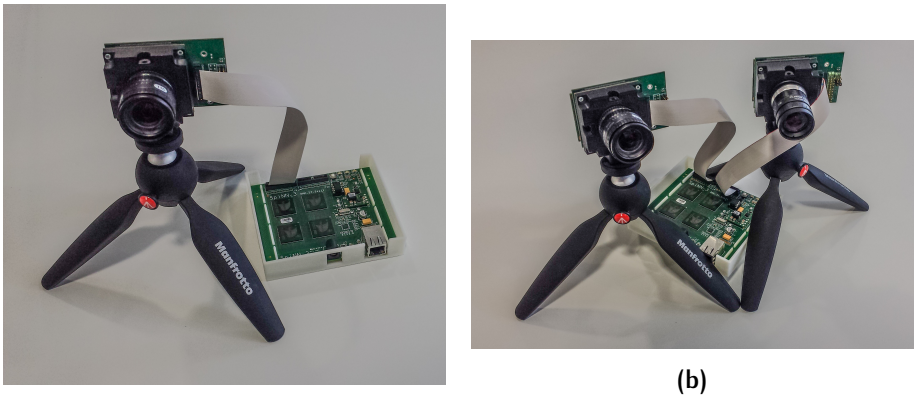


Figure 35 – SpiNNaker packet structure and event to MC packet mapping.

Injecting spikes from a single ATIS silicon retina into SpiNNaker follows the same technique presented in [37], they both rely on using the link provided in chip 0,0 for both boards. A region of interest of 128×128 pixels is selected from the original ATIS sensor space, and are assigned a virtual chip 254,254 and injected in chip 0,0. During the mapping process extra neural applications, called *Proxy*, are responsible for the translation from the virtual routing key (254,254) to a key which is physically present in the same chip where the *Proxy* application is loaded, generally chip 0,0. After the packet translation, the AER packet containing the address of the event can be routed in the SpiNNaker system as any other MultiCast packet.



(a) (b)

Figure 36 – Designed interface board plugged between an ATIS camera (a) and the SpiNNaker system, or two ATIS cameras (b) to the SpiNN3 platform.

6.4 INTERFACING TWO EVENT-BASED CAMERAS TO SPINNAKER

6.4.1 SpiNN3 board

Connecting two Atis on the 4-chip SpiNNaker board was developed with the same approach : the two available SpiNNaker links on the 4-chip board were used to inject spikes. Figure 36.b shows the two cameras connected to the SpiNN3 board. The models described in sections 6.5 and 6.6 require more than the 10k neurons available on the SpiNN3 platform (see Figure 38b). Moreover, the UDP link (over ethernet) for retrieving spikes has a bandwidth limited to 1Mbits/s. With our 16bit AER events, we can output 64kev/s, which is way lower than the event rate of a single camera (~ 300 kev/s in standard cases). Then, we need to upgrade to a bigger board in order to achieve the desired resolution and live streaming of data.

6.4.2 SpiNN5 board

Moving to the SpiNN5 platform (42 chips) required to develop a new interface for the two cameras : this board does not have a dual SpiNNaker link, but the SpiNN3 platform is not capable of simulating our networks.

As shown in Figure 38b, the number of neurons required to compute the disparity map is higher than the one a SpiNN3 board can handle. Work from [140] was used in order to achieve this: an AERNode board receives events from both cameras, merge them and send them to the SpiNNaker board through the SATA bidirectional connection (Xilink High Speed LVDS connection protocol AURORA [52]). The AER protocol was used to inject events through SpiNN5 FPGAs [99]. The F1-Lo1 link (chip 0, 0, link West from FPGA F1 - Figure 38a) was used to inject the spikes in the platform. Such a bus reaches up to 2Mev/s. As the ethernet outgoing link was limited to 64 kev/s (one UDP packet per event per millisecond), the same link was used to send the events back to the AERNode Board[139], and then back to the computer through an AER-USB mini2 board[12]. Figure 37 shows two ATIS sensors connected to the Node board and then going to the SpiNNaker platform via a SATA link.

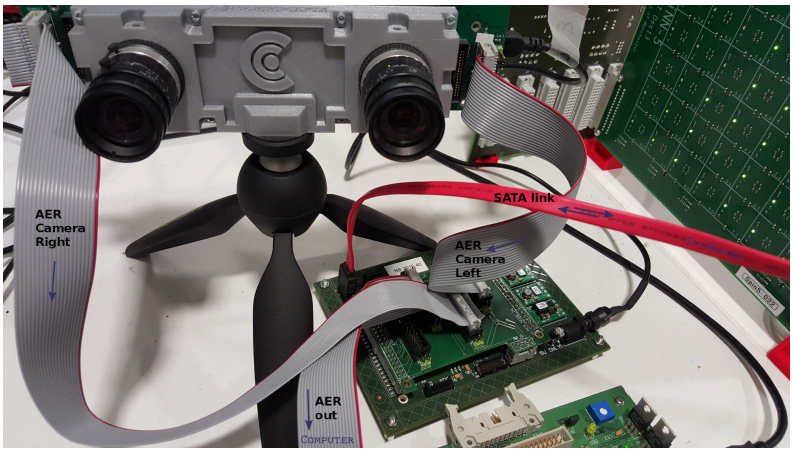


Figure 37 – Stereo setup. Two ATIS sensors, with AER connection to the Node Board, then packed in the Sata link to SpiNNaker

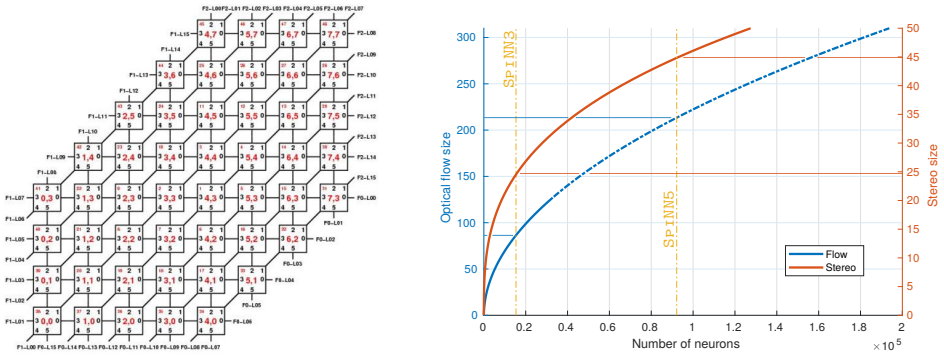


Figure 38 – (a) SpiNN5 hardware description and FPGAs links .(b) Number of neurons needed for presented networks, as a function of the number of pixel. Two limits are represented : the small 4 chips SpiNN3 board, and the big 48 chips SpiNN5 board.

6.5 OPTICAL FLOW

The full description of the model can be found in Chapter 5.

Our complete model comprises $64 \times 64 \times 4$ direction sensitive units for a total of 64×64 motion detectors. Each motion detector receives input from a 2×2 macropixel of the original retinal resolution (128×128) through a subsampling population which acts as a robust edge detector. Figure 39 shows the output of the network. On the SpiNN3 platform, we can implement up to a 86×86 network, 213×213 on the SpiNN5 (see Figure 38b).

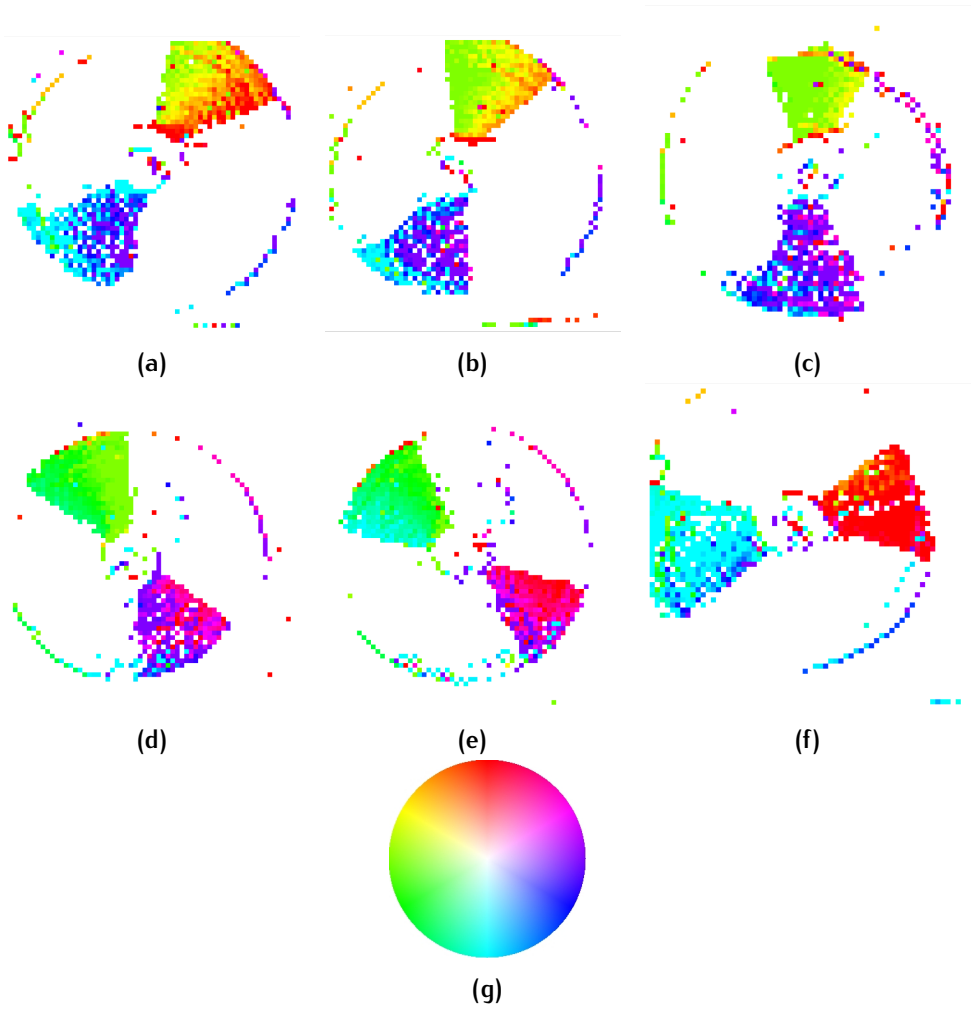


Figure 39 – Output of the flow network for a Counter Clockwise rotation of a pen (a to f). Direction is encoded among the HSV map given in (g). The overall direction is the expected one, even if some noise events trigger false computations.

6.6 DISPARITY DETECTOR

Detecting temporal coincidence between two spikes is a widely used feature in spiking neural networks. As a consequence, we introduce a novel neuron model for SpiNNaker, as a dedicated core for this task instead of using standard integrate and fire neurons which would introduce an un-

necessary overhead. Each neuron simulated by this core has two types of synaptic input and a time window. When an incoming spike is received on one input, the neuron will output a spike if another spike was received on its second input in the given time window. We added a refractory period to this process to limit the maximum firing rate of the neurons if required.

We perform stereovision using the asynchronous high temporal resolution properties of the ATIS camera as developed in [111]. The combination of spatial and temporal constraints fully exploits the high temporal resolution of neuromorphic retinas. It allows us to produce a stereo algorithm that is able to perform stereo computation by detecting coactive pixels laying on the same epipolar lines. Figure 40 shows a simplified version of the network, reduced to the matching between to epipolar lines x_l and x_r. The input neurons connect to the synchrony detectors in an excitatory fashion. A Full description of the network can be found in [94]. In our setup, as we are subsampling the input space by a factor of 4, we assume that the epipolar lines are lying on the horizontal axis of the camera, thus allowing us to skip the rectification process. Special care has been taken during the mechanical assembly to ensure the parallelism of the two sensors.

The hereby described networks required $S_x \times S_y^2$ neurons, where S_x and S_y are respectively the x and y sizes of the subsampled population. 32768 neurons are required for a $S_x = 32$, $S_y = 32$ population. The SpiNN3 board can contain up to a 24×24 network, 44×44 for the SpiNN5 (see Figure 38b).

The same experiment is repeated by taking inputs from the two cameras and feeding them on the SpiNNaker in real-time. Figure 41 shows different snapshots of the network output, obtained by waving a pen in front of the cameras. We show the events coming from each silicon retina and the result of the disparity computation in a 32×32 plane. The color encodes the disparity value: a big disparity, or, in other words, a stimulus being close to the camera corresponds to red colors (Figure 41.c). Likewise, yellowish colors correspond to a lower level of disparity, representing an object further away (Figure 41.a).

6.7 CONCLUSION

The research presented in this chapter introduces the first stages of a full neuromorphic pipeline, from an event-based camera to a fully asynchronous spiking neural network dedicated hardware. The first contribution consists in the introduction of the novel interface between our event-based sensors and the SpiNNaker platform. Most of the work found in the literature considered in pre-loading data on the platform (and thus preventing real-time applications), combined with a reduced network size in order to fit in a single board, as the bandwidth of the on-board ethernet is limited. To get beyond small, proof-of-concept models, we wanted to connect two 128×128 pixels sensor, with an event rate up to 1Meps for the two cameras. The developed architecture is able to both inject and retrieve up to 4Meps in the SpiNNaker, from two incoming AER nodes. A subsequent contribution

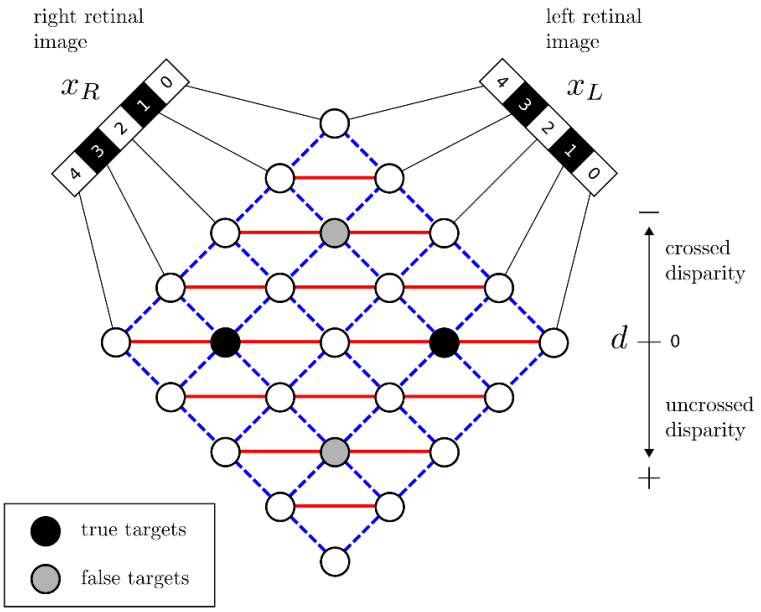


Figure 40 – The proposed cooperative network for stereo matching, as described in [73]. The output from the two epipolar lines x_l and x_r both connects to the synchrony detectors. The blue lines represent excitatory links, as the red ones inhibitory links. The final output encodes a representation of the original scene in disparity space. For the sake of visibility, only a horizontal line of retinal cells is considered. The corresponding coincidence and disparity detector units, hence, lie within a horizontal plane. Again, only few units are shown here, while in the complete network units would be uniformly distributed over the entire plane.

consists in mapping the Optical Flow and Stereo Vision algorithms on the SpiNNaker platform. The current SpiNN5 board allows us to implement a 213×213 pixels optical flow. The main issue that we faced was the instability of the SpiNNaker platform. As we are injecting and retrieving spikes from a virtual core, a single physical core sees all the events going through its router, making it randomly crashing. The problem may come either from the centralized message passing structure or the (legacy) software toolchain used, or both. Due to this instability, we weren't able to benchmark the provided algorithms. Some effort was put in order to move to the new software toolchain provided by University of Manchester, and first encouraging results were obtained. More, as the University of Dresden is designing the new SpiNNaker chip, I was able to try some implementations on the new SpiNNaker2 chip, which seems promising for the future as a single chip will have the computational abilities of the current SpiNN5 board.

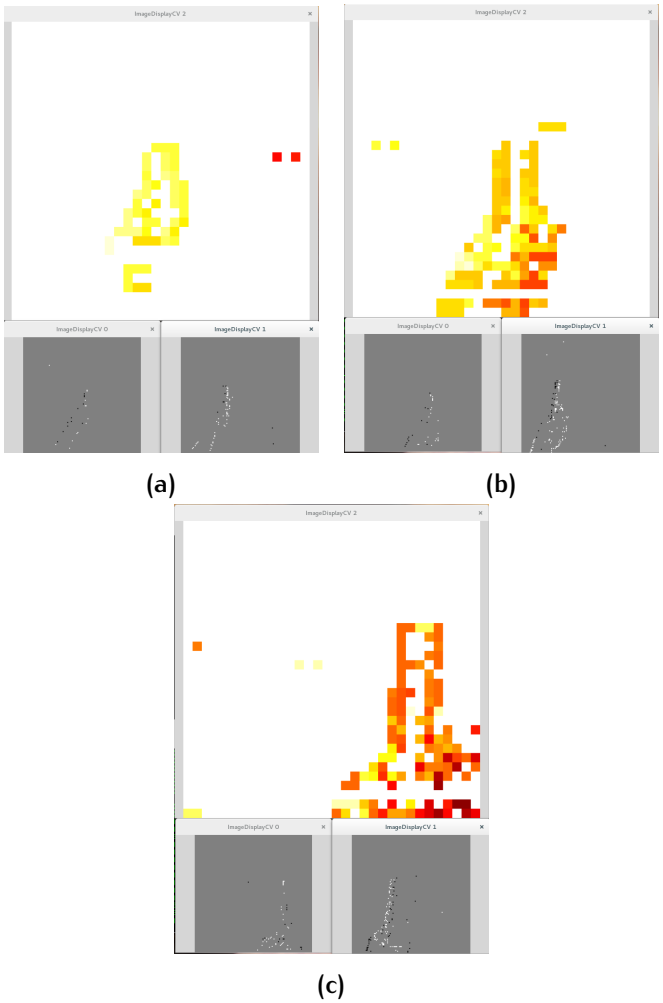


Figure 41 – Results of disparity computation with real-time input from two ATIS. A stimulus far from the camera will produce a small disparity, encoded in yellowish colors, a close one will produce a big disparity, encoded in red color.

7 CONCLUSION

This thesis entitled *Neuromorphic computation using event-based sensors : from algorithms to hardware implementations*, is made of a voluntary ambiguity. The *neuromorphic computation* **using** *event-based sensors* reflects that computation can be either done *by* the sensor itself, or *with* data coming from this kind of sensors. As previously stated, the event-driven sensors offer a huge dynamic range, an accurate temporal resolution and a sensor-level data compression. All along this manuscript, we tried to explore the precise timing of the event stream, avoiding binarization or events binning, both methods leading to the classical framework of computing vision : Frames.

HINDSIGHTS

DEFOCUS In Chapter 2, we used the precise timing between two consecutive events with opposite polarity, generated by an event-based camera and a liquid lens, to extract, in real time, a depth map of the scene. The control of the liquid lens and the sensitivity of the sensor is the actual limitation for this algorithm. With available technologies, we were able to compute such a depth map at 100Hz, which is, as far as we know, the highest refresh rate reported so far for a monocular system.

SPARSE CODING Chapter 3 presented an evolution of the original HOTS algorithm, introducing sparse coding methods to reduce the number of prototypes used for spatiotemporal contexts clustering. This can be of a huge advantage when thinking about hardware implementation, as the number of neurons available on state-of-the-art neuromorphic hardware is low. But this reduction of prototypes has an increasing impact on the number of spikes generated, which can be a problem on some specific hardware, such as the SpiNNaker platform : too much activity can crash some cores of the platform. This method can not be suitable, in its initial version, for online learning, as we need to get all the spatiotemporal contexts before running the optimization process.

SPIKE SORTING The spike sorting method presented in Chapter 4 uses the timing of consecutive spikes to extract low level features in a hierarchical way. This work has proven its ability to tackle state-of-the-art algorithms, while running in real-time, and is a good candidate for hardware implementation. Further work will have to quantify the robustness of this algorithm to limited precision and noise. If it can be shown that this algorithm stays stable towards noise and quantization, I do believe this is a very good candidate for smarter brain-machine interfaces.

FLOW ON TRUENORTH The 22nd edition of the Telluride Neuromorphic Workshop gave me the possibility to implement an optical flow algorithm on TrueNorth, IBM's Neurosynaptic plaform (Chapter 5). This method uses timing of spikes in a close surrounding to infer the direction of motion in the scene. To some extent, this work could be strengthened to noise and false alarm but shaping the excitation and inhibition neurons in circle, in the manner of the ganglion cells receptive field.

SPINNAKER Chapter 6 was the first project I was involved in during my PhD. Here, again, the timing of the spikes is of a crucial importance for the disparity matcher, as we are looking for co-activation of two pixels among the same epipolar line in both cameras. Unfortunately, the SpiNNaker platform and the software pipeline were not reliable enough in order to be able to fully characterize the implemented networks. It was almost impossible to launch twice the same simulation without crashing a core of the SpiNNaker. This is, I guess, inherent to the architecture topology, as we are flooding a single core for inputing/outputting spikes to the platform.

PERSPECTIVES

Wijekoon et. Al proposed an VLSI implementation of the Izhikevich neuron model[136]. This model does not fully replicate the behavioral equations, but reflects the trend of the neuron's state (membrane potential, synaptic current) to change towards a given input. There is something beautiful in this paper : that we do not always have to fit (and stick) to behavioral equations, but rather to analyze behavior and use available knowledge to build something that behaves in the same way. Thinking this way is an ability that I hope I acquired during this PhD, and I hope I will have the opportunity of applying this during the next years.

All among this thesis, we tried to be as fair as possible by comparing the proposed method to state-of-the-art algorithms. Unfortunately, due to the nature of the signals we are dealing with, it is a hard challenge to directly compare results coming from an event-based algorithm to their standard machine learning/computer vision counterparts. The performances of the presented methods in this manuscript were obtained with known/labeled datasets, when available. As the spike sorting community is willing to get more powerful analysis tools, there are a lot of synthetic or recorded labeled databases. This was a great help when developing/tweaking the algorithms. Unfortunately, this kind of database is not all the time available, especially in the neuromorphic community. A Real(ly) event-based dataset should, in my opinion, be developed, so as to unify the results and harmonize the comparison in between the proposed algorithms. To be completely fair, this database should contain the same stimulus, recorded both with an event-based sensor and a regular sensor, thus allowing the whole community to compete, and then validate the superiority of event-based approaches.

The end of my PhD drove me towards thinking about brain-machine interfaces. Decoding, in real-time, cerebral activity is nowadays an open issue.

Prior work [138] has shown that it is possible to decode neural activity in the human brain, after a daily recalibration of the system that takes a significant amount of time, thus preventing this system to be widely used. Introducing a paradigm shift in the data representation can lead to a significant gain in both performances and power consumption. A last chapter could have been given, introducing some interesting thoughts about the possible VLSI implementation of the spike sorting algorithm. It has been a chance to discover this totally new field for me, and also a way to understand more thoroughly the behavior of such systems (including our silicon retina), and the reason why of all the biases in such sensors. Future work could be achieved in order to fully characterize the proposed hardware implementation. Moreover, there were some issues with the MEA chip, preventing us to get recordings to work with. Merging the MEA chip with some low-level computation, for embedded, smarter, brain-machine interfaces, could be an interesting project to conduct...

This achieves more than 3 years of reflection. I had the chance to cover almost all the neuromorphic spectrum, from algorithms to hardware implementation and design. Algorithms are entering a maturity phase : the community now understands that there is a need (and benefit) to use the precise timing for computation purposes. Even if there is a theoretical gap, as we do not have any mathematical formulation as Shannon expressed one, there is hope to achieve major breakthrough in the next years or decades. Nevertheless, standard computers that we are using as computation platforms for our algorithms are not the most efficient ones, as most of the proposed algorithms work in a sequential manner, thus preventing to efficiently use the many core architecture which is nowadays present in all processors. This is why the mixed-signal community will be of a crucial importance. In the very beginning of my thesis, I struggled understanding the biases of the silicon retina. Later on, I understood that subthreshold transistors are very sensitive animals, corrupted by mismatch and noise, and that building a (working) silicon chip is a hard process. However, every new neuromorphic chip outperforms the previous one. Research is going its way, and tomorrow will see the growth of new architectures (either spiking or sequential ones), that will allow the community to build powerful embedded applications.
All among these years, I had to go through the process of reading and writing papers. One would open the question of the reviewing scheme. Is this scheme not from an old and ancient time ? Do we really need peer-reviewed papers, with all the problems of conflicts it brings ? I was also amazed by the cost of publishing a paper. Is there an equity there ? Some small labs could have good ideas, but won't be able to publish them because they do not have the funding for. Where does this money go, as the reviewers are not paid ? This is maybe the most expensive layout I ever saw... Shouldn't all the papers be in open access ? Or even free to publish, with a voting system increasing the impact of the best ones ?

Achieved on August 24, 2018.

Figure 42 – Chart of this thesis' words count, generated with *www.wordclouds.com*

BIBLIOGRAPHY

[1] S. Afshar, L. George, J. Tapson, A. van Schaik, and T. J. Hamilton. « Racing to learn: statistical inference and learning in a single spiking neuron with adaptive kernels ». In: *Frontiers in neuroscience* 8 (2014), p. 377 (cit. on p. 31).

[2] L. Alvarez, J. Sánchez, and J. Weickert. « A scale-space approach to nonlocal optical flow calculations ». In: *International conference on scale-space theories in computer vision*. Springer. 1999, pp. 235–246 (cit. on p. 41).

[3] A. Amir, P. Datta, W. P. Risk, A. S. Cassidy, J. A. Kusnitz, S. K. Esser, A. Andreopoulos, T. M. Wong, M. Flickner, R. Alvarez-Icaza, et al. « Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores ». In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE. 2013, pp. 1–10 (cit. on p. 46).

[4] T. Baden, F. Schaeffel, and P. Berens. « Visual Neuroscience: A Retinal Ganglion Cell to Report Image Focus?. » In: *Curr. Biol.* 27 (2017), pp. 138–141 (cit. on p. 20).

[5] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. « A database and evaluation methodology for optical flow ». In: *International Journal of Computer Vision* 92.1 (2011), pp. 1–31 (cit. on p. 54).

[6] W. Barbakh and C. Fyfe. « Online clustering algorithms ». In: *International Journal of Neural Systems* 18.03 (2008), pp. 185–194 (cit. on pp. 22, 33, 34).

[7] P. Bardow, A. J. Davison, and S. Leutenegger. « Simultaneous optical flow and intensity estimation from an event camera ». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 884–892 (cit. on pp. 42, 59).

[8] H. B. Barlow and W. R. Levick. « The mechanism of directionally selective units in rabbit's retina. » In: *The Journal of physiology* 178.3 (1965), p. 477 (cit. on pp. 41–43, 61).

[9] F. Barranco, C. Fermüller, and Y. Aloimonos. « Contour motion estimation for asynchronous event-driven cameras ». In: *Proceedings of the IEEE* 102.10 (2014), pp. 1537–1556 (cit. on pp. 42, 59).

[10] I. Barron. « D.. I. Fleet, and SS Beauchemin, Performance of optical flow techniques ». In: *International Journal of Computer Vision* 12.2 (1994) (cit. on p. 41).

[11] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan. « Asynchronous frameless event-based optical flow ». In: *Neural Networks* 27 (2012), pp. 32–37 (cit. on pp. 42, 59).

[12] R. Berner, T Delbruck, A Civit-Balcells, and A. Linares-Barranco. « A 5 Meps $100 USB2. 0 address-event monitor-sequencer interface ». In: *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE. 2007, pp. 2451–2454 (cit. on p. 65).

[13] M. J. Berry, D. K. Warland, and M. Meister. « The structure and precision of retinal spike trains ». In: *Proceedings of the National Academy of Sciences* 94.10 (1997), pp. 5411–5416. ISSN: 0027-8424. DOI: 10.1073/pnas.94.10.5411. eprint: http://www.pnas.org/content/94/10/5411.full.pdf (cit. on p. 5).

[14] A. Bhattacharyya. « On a measure of divergence between two multinomial populations ». In: *Sankhyā: the indian journal of statistics* (1946), pp. 401–406 (cit. on pp. 26, 35).

[15] J. Binas, G. Indiveri, and M. Pfeiffer. « Spiking analog VLSI neuron assemblies as constraint satisfaction problem solvers ». In: *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE. 2016, pp. 2094–2097 (cit. on p. 6).

[16] M Blum, M Büeler, C Grätzel, and M Aschwanden. « Compact optical design solutions using focus tunable lenses ». In: *SPIE Optical Systems Design*. International Society for Optics and Photonics. 2011, 81670W–81670W (cit. on p. 12).

[17] K. Boahen. « Point-to-point connectivity between neuromorphic chips using address events ». English. In: *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.* 47.5 (2000), pp. 416–434. ISSN: 10577130. DOI: 10.1109/82.842110 (cit. on p. 51).

[18] T. Brosch and H. Neumann. « Event-based optical flow on neuromorphic hardware ». In: *proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS) on 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 2016, pp. 551–558 (cit. on p. 42).

[19] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, et al. « Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores ». In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE. 2013, pp. 1–10 (cit. on pp. 46, 47, 50).

[20] T. N. Chandrapala and B. E. Shi. « The generative adaptive subspace self-Organizing Map ». In: *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE. 2014, pp. 3790–3797 (cit. on p. 31).

[21] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri. « Neuromorphic electronic circuits for building autonomous cognitive systems. » In: *Proceedings of the IEEE* 102 (2014), pp. 1367–1388 (cit. on p. 6).

[22] K. Ciuffreda. « Why two eyes? » In: 13. 2002, pp. 35–37 (cit. on p. 7).

[23] K. Ciuffreda and K. Engber. « Is one eye better than two when viewing pictorial art? » In: 35. 2002, pp. 37–40 (cit. on p. 7).

[24] G. K. Cohen, G. Orchard, S.-H. Leng, J. Tapson, R. B. Benosman, and A. Van Schaik. « Skimming digits: neuromorphic classification of spike-encoded images ». In: *Frontiers in neuroscience* 10 (2016), p. 184 (cit. on p. 31).

[25] F. Corradi and G. Indiveri. « A neuromorphic event-based neural recording system for smart brain-machine-interfaces ». In: *IEEE transactions on biomedical circuits and systems* 9.5 (2015), pp. 699–709 (cit. on p. 40).

[26] T. Cover and P. Hart. « Nearest neighbor pattern classification ». In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27 (cit. on p. 34).

[27] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang. « Loihi: A Neuromorphic Manycore Processor with On-Chip Learning ». In: *IEEE Micro* 38.1 (2018), pp. 82–99. ISSN: 0272-1732. DOI: 10.1109/MM.2018.112130359 (cit. on pp. 6, 20).

[28] 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 ». In: *Frontiers in neuroinformatics* 2 (2008) (cit. on p. 15).

[29] T. Delbrück. « Silicon retina with correlation-based, velocity-tuned pixels ». In: *IEEE Trans. on Neural Networks* 4.3 (1993), pp. 529–541. ISSN: 1045-9227. DOI: 10.1109/72.217194 (cit. on p. 42).

[30] G. Dikov, M. Firouzi, F. Röhrbein, J. Conradt, and C. Richter. « Spiking Cooperative Stereo-Matching at 2 ms Latency with Neuromorphic Hardware ». In: *Conference on Biomimetic and Biohybrid Systems*. Springer. 2017, pp. 119–137 (cit. on p. 6).

[31] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. « Flownet: Learning optical flow with convolutional networks ». In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2758–2766 (cit. on p. 41).

[32] S. Esser, P. Merolla, J. Arthur, A. Cassidy, R Appuswamy, A Andreopoulos, D. Berg, J. McKinstry, T Melano, D. Barch, et al. « Convolutional networks for fast, energy-efficient neuromorphic computing. 2016 ». In: *Preprint on ArXiv. http://arxiv. org/abs/1603.08270. Accessed* 27 (2016) (cit. on p. 3).

[33] S. Fisher and K. Ciuffreda. « Accommodation and apparent distance ». In: 17. 1988, pp. 609–621 (cit. on p. 7).

[34] C. Frenkel, J.-D. Legat, and D. Bol. « A 0.086-mm2 9.8-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28nm CMOS ». In: *arXiv preprint arXiv:1804.07858* (2018) (cit. on p. 3).

[35] S. Furber, F. Galluppi, S. Temple, and L. Plana. « The spinnaker project ». In: *Proceedings of the IEEE* 102.5 (2014), pp. 652–665 (cit. on pp. 6, 15, 20, 62).

[36] V. Gaganov and A. Ignatenko. « Robust shape from focus via Markov random fields ». In: *Proceedings of Graphicon Conference*. 2009, pp. 74–80 (cit. on p. 7).

[37] F Galluppi, C Denk, M. C. Meiner, T Stewart, L. A. Plana, C Eliasmith, S Furber, and J Conradt. « Event-based neural computing on an autonomous mobile platform ». In: *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 1–6 (cit. on p. 64).

[38] J. A. Gasthaus. « Spike sorting using time-varying DIRICHLET process mixture models ». PhD thesis. Citeseer, 2008 (cit. on p. 40).

[39] M.-O. Gewaltig and M. Diesmann. « Nest (neural simulation tool) ». In: *Scholarpedia* 2.4 (2007), p. 1430 (cit. on p. 15).

[40] S. Ghosh-Dastidar and H. Adeli. « Spiking neural networks ». In: *International journal of neural systems* 19.04 (2009), pp. 295–308 (cit. on p. 6).

[41] M. Giulioni, X. Lagorce, F. Galluppi, and R. B. Benosman. « Event-based computation of motion flow on a neuromorphic analog neural platform ». In: *Frontiers in neuroscience* 10 (2016) (cit. on pp. 6, 43–45, 59).

[42] T. Gollisch and M. Meister. « Rapid Neural Coding in the Retina with Relative Spike Latencies ». In: *Science* 319.5866 (2008), pp. 1108–1111. ISSN: 0036-8075. DOI: 10.1126/science.1149639. eprint: http://science.sciencemag.org/content/319/5866/1108.full.pdf (cit. on p. 5).

[43] V. Grant. « Accommodation and convergence in visual space perception ». In: 31. 1942, pp. 89–104 (cit. on p. 7).

[44] G. Haessig and X. Berthelon. *https://youtu.be/Ia5gfVLnoaY*. Youtube. 2017. URL: https://youtu.be/Ia5gfVLn0aY (cit. on p. 15).

[45] G. Haessig, A. Cassidy, R. Alvarez, R. Benosman, and G. Orchard. « Spiking Optical Flow for Event-based Sensors Using IBM's TrueNorth Neurosynaptic System ». In: *IEEE Transactions on Biomedical Circuits and Systems* 99 (2018), pp. 1–11 (cit. on p. 6).

[46] K. D. Harris, D. A. Henze, J. Csicsvari, H. Hirase, and G. Buzsáki. « Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements ». In: *Journal of neurophysiology* 84.1 (2000), pp. 401–414 (cit. on p. 39).

[47] D. J. Heeger. « Model for the extraction of image flow ». In: *JOSA A* 4.8 (1987), pp. 1455–1471 (cit. on p. 41).

[48] R. T. Held, E. A. Cooper, J. F. O'Brien, and M. S. Banks. « Using Blur to Affect Perceived Distance and Size ». In: *ACM Transactions on Graphics* 29.2 (2010), 19:1–16. DOI: 10.1145/1731047.1731057 (cit. on p. 5).

[49] D Henze, K. Harris, Z Borhegyi, J Csicsvari, A Mamiya, H Hirase, A Sirota, and G Buzsáki. *Simultaneous intracellular and extracellular recordings from hippocampus region CA1 of anesthetized rats*. 2009 (cit. on pp. 31, 39).

[50] D. A. Henze, Z. Borhegyi, J. Csicsvari, A. Mamiya, K. D. Harris, and G. Buzsáki. « Intracellular features predicted by extracellular recordings in the hippocampus in vivo ». In: *Journal of neurophysiology* 84.1 (2000), pp. 390–400 (cit. on p. 39).

[51] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys. « An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications ». In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 1736–1741 (cit. on p. 41).

[52] X. Inc. *Aurora 8B/10B Protocol Specification*. Tech. rep. 2011 (cit. on p. 65).

[53] G. Indiveri, F. Corradi, and N. Qiao. « Neuromorphic architectures for spiking deep neural networks. » In: *IEEE Electron Devices Meeting (IEDM)* (2015), pp. 1–4 (cit. on p. 6).

[54] M. Joesch, B. Schnell, S. V. Raghu, D. F. Reiff, and A. Borst. « ON and OFF pathways in Drosophila motion vision ». In: *Nature* 468.7321 (2010), p. 300 (cit. on p. 41).

[55] R. Johansson and I. I. Birznieks. « First spikes in ensembles of human tactile afferents code complex spatial fingertip events. » In: *Nat Neurosci* 7 (2004), pp. 170–177 (cit. on p. 6).

[56] S. N. Kadir, D. F. Goodman, and K. D. Harris. « High-dimensional cluster analysis with the masked EM algorithm ». In: *Neural computation* (2014) (cit. on p. 31).

[57] K. Khoshelham. « Accuracy analysis of kinect depth data ». In: *ISPRS workshop laser scanning*. 2011, pp. 133–138 (cit. on p. 16).

[58] X. Lagorce and R. Benosman. « Stick: Spike time interval computational kernel, a framework for general purpose computation using neurons, precise timing, delays, and synchrony ». In: *Neural computation* (2015) (cit. on p. 25).

[59] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman. « Hots: a hierarchy of event-based time-surfaces for pattern recognition ». In: *IEEE transactions on pattern analysis and machine intelligence* 39.7 (2017), pp. 1346–1359 (cit. on pp. 21, 23–28, 30, 31, 39).

[60] L. Lapicque. « Recherches quatitatives sur l'excitation electrique des nerfs traitee comme polarisation ». In: *J. Physiol. Pathol. Gen.* 9 (1907), pp. 620–635 (cit. on p. 13).

[61] Y. LeCun, Y. Bengio, et al. « Convolutional networks for images, speech, and time series ». In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995 (cit. on p. 3).

[62] J. Y. Lettvin, H. R. Maturana, W. S. McCulloch, and W. H. Pitts. « What the frog's eye tells the frog's brain ». In: *Proceedings of the IRE* 47.11 (1959), pp. 1940–1951 (cit. on p. 41).

[63] A. Levin, R. Fergus, F. Durand, and W. Freeman. « Image and depth from a conventional camera with a coded aperture ». In: vol. 26. 3. 2007, pp. 1–70 (cit. on p. 7).

[64] H.-Y. Lin and C.-H. Chang. « Depth recovery from motion and defocus blur ». In: *Image Analysis and Recognition* (2006), pp. 122–133 (cit. on p. 9).

[65] R. C. Liu, S. Tzonev, S. P. Rebrik, and K. D. Miller. « Variability and information in a neural code of the cat lateral geniculate nucleus. » In: *Journal of neurophysiology* 86 6 (2001), pp. 2789–806 (cit. on p. 6).

[66] W. Maass. « Computing with spiking neurons ». In: *Pulsed neural networks* 85 (1999) (cit. on p. 6).

[67] L. v. d. Maaten and G. Hinton. « Visualizing data using t-SNE ». In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605 (cit. on p. 36).

[68] J. MacQueen. « Some methods for classification and analysis of multivariate observations ». In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297 (cit. on p. 34).

[69] R. Macknojia, A. Chavez-Aragon, P. Payeur, and R. Laganiere. « Experimental characterization of two generations of Kinect's depth sensors ». In: *IEEE International symposium on robotic and sensor environements*. 2012. DOI: 10.1109/ROSE.2012.6402634 (cit. on p. 16).

[70] M. Mahowald. « VLSI analogs of neuronal visual processing: a synthesis of form and function ». PhD thesis. California Institute of Technology, 1992 (cit. on p. 2).

[71] Z. Mainen and T. Sejnowski. « Reliability of spike timing in neocortical neurons. » In: *Science* 268 (1995), pp. 1503–1506 (cit. on p. 6).

[72] A. Mani and G. Schwartz. « Circuit mechanisms of a retinal ganglion cell with stimulus-dependent response latency and activation beyond its dendrites. » In: *Curr. Biol.* 27 (2017), pp. 471–482 (cit. on p. 20).

[73] D. Marr and T. Poggio. « Cooperative computation of stereo disparity ». In: *Science* 194.4262 (1976), pp. 283–287 (cit. on pp. 61, 62, 68).

[74] J. N. Martel, L. K. Müller, S. J. Carey, and P. Dudek. « High-Speed Depth from Focus on a Programmable Vision Chip Using a Focus Tunable Lens ». In: *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE. 2017, pp. 1150–1153 (cit. on p. 7).

[75] J. M. Mateos-Pérez, R. Redondo, R. Nava, J. C. Valdiviezo, G. Cristóbal, B. Escalante-Ramírez, M. J. Ruiz-Serrano, J. Pascau, and M. Desco. « Comparative evaluation of autofocus algorithms for a real-time system for automatic detection of Mycobacterium tuberculosis ». In: *Cytometry Part A* 81.3 (2012), pp. 213–221 (cit. on p. 7).

[76] G. Mather. « Image blur as a pictorial depth cue ». In: 263. 1996, pp. 169–172 (cit. on p. 7).

[77] G. Mather. « The use of image blur as a depth cue ». In: 26. 1997, pp. 1147–1158 (cit. on p. 7).

[78] G. Mather and D. Smith. « Blur discrimination and its relation to blur-mediated depth perception ». In: 31. 2002, pp. 1211–1219 (cit. on p. 7).

[79] G. Mather and D. Smith. « Combining depth cues: effects upon accuracy and speed of performance in a depth-ordering task ». In: 44. 2004, pp. 557–562 (cit. on p. 7).

[80] G. Mather and D. Smith. « Depth cue integration: stereopsis and image blur ». In: 40. 2000, pp. 3501–3506 (cit. on p. 7).

[81] C. A. Mead and M. A. Mahowald. « A silicon model of early visual processing ». In: *Neural networks* 1.1 (1988), pp. 91–97 (cit. on p. 2).

[82] 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, et al. « A million spiking-neuron integrated circuit with a scalable communication network and interface ». In: *Science* 345.6197 (2014), pp. 668–673 (cit. on pp. 6, 15, 20, 45).

[83] P. Merolla, R. Appuswamy, J. Arthur, S. K. Esser, and D. Modha. « Deep neural networks are robust to weight binarization and other non-linear distortions ». In: *arXiv preprint arXiv:1606.01981* (2016) (cit. on p. 45).

[84] M. Moeller, M. Benning, C. Schönlieb, and D. Cremers. « Variational depth from focus reconstruction ». In: *IEEE Transactions on Image Processing* 24.12 (2015), pp. 5369–5378 (cit. on p. 7).

[85] G. E. Moore et al. « Progress in digital integrated electronics ». In: *Electron Devices Meeting*. Vol. 21. 1975, pp. 11–13 (cit. on p. 2).

[86] H. Mostafa, L. K. Müller, and G. Indiveri. « An event-based architecture for solving constraint satisfaction problems ». In: *Nature communications* 6 (2015), p. 8941 (cit. on p. 6).

[87] J. Navaridas, S. Furber, J. Garside, X. Jin, M. Khan, D. Lester, M. Luján, J. Miguel-Alonso, E. Painkras, C. Patterson, et al. « Spinnaker: fault tolerance in a power-and area-constrained large-scale neuromimetic architecture ». In: *Parallel Computing* 39.11 (2013), pp. 693–708 (cit. on p. 62).

[88] E. e. a. Neftci. « Synthesizing cognition in neuromorphic electronic systems. » In: *Proceedings of the National Academy of Sciences* (2013), pp. 3468–3476 (cit. on p. 6).

[89] V. Nguyen, I. Howard, and R. Allison. « Detection of the depth order of defocused images ». In: 45. 2005, pp. 1003–1011 (cit. on p. 7).

[90] B. A. Olshausen et al. « Emergence of simple-cell receptive field properties by learning a sparse code for natural images ». In: *Nature* 381.6583 (1996), pp. 607–609 (cit. on pp. 23, 24).

[91] G. Orchard, R. Benosman, R. Etienne-Cummings, and N. V. Thakor. « A spiking neural network architecture for visual motion estimation ». In: *Biomedical Circuits and Systems Conference (BioCAS), 2013 IEEE*. IEEE. 2013, pp. 298–301 (cit. on pp. 42, 52).

[92] G. Orchard and R. Etienne-Cummings. « Bioinspired visual motion estimation ». In: *Proceedings of the IEEE* 102.10 (2014), pp. 1520–1536 (cit. on pp. 42, 52).

[93] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman. « HFirst: a temporal approach to object recognition ». In: *IEEE transactions on pattern analysis and machine intelligence* 37.10 (2015), pp. 2028–2040 (cit. on pp. 21, 27).

[94] M. Osswald, S.-H. Ieng, R. Benosman, and G. Indiveri. « A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems ». In: *Scientific reports* 7 (2017), p. 40703 (cit. on pp. 6, 67).

[95] A. Pentland. « A new sense for depth of field. » In: 9. 1987, pp. 523–531 (cit. on p. 7).

[96] A. Pentland, S. Scherock, T. Darrel, and B. Girod. « Simple range cameras based on focal error. » In: vol. 11. 1. 1994, pp. 2925–2934 (cit. on p. 7).

[97] A. P. Pentland. « A new sense for depth of field ». In: *IEEE transactions on pattern analysis and machine intelligence* 4 (1987), pp. 523–531.

[98] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco. « Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate Coding and Coincidence Processing–Application to Feedforward ConvNets ». In: *IEEE transactions on pattern analysis and machine intelligence* 35.11 (2013), pp. 2706–2719 (cit. on pp. 21, 27).

[99] L. A. Plana. *AppNote 8 - Interfacing AER devices to SpiNNaker using an FPGA*. Tech. rep. Manchester: University of Manchester, 2013 (cit. on pp. 63, 65).

[100] C. Posch, D. Matolin, and R. Wohlgenannt. « A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS ». In: *IEEE Journal of Solid-State Circuits* 46.1 (2011), pp. 259–275 (cit. on pp. 22, 32, 45, 61).

[101] C. Posch, D. Matolin, and R. Wohlgenannt. « High-dr frame-free pwm imaging with asynchronous aer intensity encoding and focal-plane temporal redundancy suppression ». In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE. 2010, pp. 2430–2433 (cit. on pp. 6, 8).

[102] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck. « Retinomorphic event-based vision sensors: bioinspired cameras with spiking output ». In: *Proceedings of the IEEE* 102.10 (2014), pp. 1470–1484 (cit. on p. 45).

[103] W. H. Press, S. Teukolsky, W. Vetterling, and B. Flannery. « Numerical recipes in C ». In: *Cambridge University Press* 1 (1988), p. 3 (cit. on p. 24).

[104] 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 ». In: *Frontiers in neuroscience* 9 (2015), p. 141 (cit. on p. 6).

[105] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul. « Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering ». In: *Neural computation* 16.8 (2004), pp. 1661–1687 (cit. on p. 31).

[106] P. R.S., S. Panzeri, and M. Diamond. « Population Coding of Stimulus Location in Rat Somatosensory Cortex ». In: *Neuron* 32 (2001), pp. 503–414 (cit. on p. 6).

[107] M Recce and J O'keefe. « The tetrode: a new technique for multi-unit extracellular recording ». In: *Soc Neurosci Abstr*. Vol. 15. 2. 1989, p. 1250 (cit. on p. 39).

[108] P. Reinagel and R. C. Reid. « Temporal Coding of Visual Information in the Thalamus ». In: *Journal of Neuroscience* 20 (Aug. 2000), pp. 5392–400 (cit. on p. 6).

[109] C. Richter, F. Röhrbein, and J. Conradt. *Bio-inspired optic flow detection using neuromorphic hardware*. Poster. 2014 (cit. on p. 42).

[110] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek. *Spikes: Exploring the Neural Code*. Cambridge, MA, USA: MIT Press, 1999. ISBN: 0-262-18174-6 (cit. on p. 6).

[111] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck. « Asynchronous event-based binocular stereo matching ». In: *IEEE Transactions on Neural Networks and Learning Systems* 23.2 (2012), pp. 347–353 (cit. on pp. 61, 62, 67).

[112] C. Rossant, S. N. Kadir, D. F. Goodman, J. Schulman, M. Belluscio, G. Buzsaki, and K. D. Harris. « Spike sorting for large, dense electrode arrays ». In: *Nat Neurosci* (2015), pp. 634–641 (cit. on p. 31).

[113] R. e. a. Serrano-Gotarredona. « CAVIAR: A 45K Neuron, 5M Synapse, 12G Connects AER Hardware Sensory-Processing- Learning-Actuating System for High-Speed Visual Object Recognition and Tracking ». In: *IEEE Transactions on Neural Networks* (2009), pp. 1417–1438 (cit. on p. 6).

[114] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. Civit Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jimenez, and B. Linares-Barranco. « CAVIAR: a 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing- learning-actuating system for high-speed visual object recognition and tracking. » In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 20.9 (2009), pp. 1417–38. ISSN: 1941-0093. DOI: 10.1109/TNN.2009.2023653 (cit. on p. 21).

[115] S. Shahid, J. Walker, and L. S. Smith. « A new spike detection algorithm for extracellular neural recordings ». In: *IEEE Transactions on Biomedical Engineering* 57.4 (2010), pp. 853–866 (cit. on p. 40).

[116] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. « Mastering the game of Go with deep neural networks and tree search ». In: *nature* 529.7587 (2016), pp. 484–489 (cit. on p. 3).

[117] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman. « HATS: Histograms of Averaged Time Surfaces for Robust Event-based Object Classification ». In: *arXiv preprint arXiv:1803.07913* (2018) (cit. on p. 31).

[118] M. Sutton, W. Wolters, W. Peters, W. Ranson, and S. McNeill. « Determination of displacements using an improved digital correlation method ». In: *Image and vision computing* 1.3 (1983), pp. 133–139 (cit. on p. 41).

[119] S. Suwajanakorn, C. Hernandez, and S. M. Seitz. « Depth from focus with your mobile phone ». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3497–3506 (cit. on p. 7).

[120] « Eye smarter than scientists believed: neural computations in circuits of the retina. » In: *Neuron* 65-2 (2010). Ed. by G. T and M. M., pp. 150–64 (cit. on p. 19).

[121] J. Tanner and C. Mead. « An integrated analog optical motion sensor ». In: (1986) (cit. on p. 42).

[122] M. W. Tao, S. Hadap, J. Malik, and R. Ramamoorthi. « Depth from Combining Defocus and Correspondence Using light-Field Cameras ». In: 2013 (cit. on p. 7).

[123] S Temple. *AppNote 1 - SpiNN-3 Development Board*. Tech. rep. Manchester: University of Manchester, 2011 (cit. on p. 63).

[124] S. J. Thorpe, A. Delorme, and R. VanRullen. « Spike-based strategies for rapid processing ». In: *NEURAL NETWORKS* 14 (2001), pp. 715–725 (cit. on p. 6).

[125] S. Thorpe. « Spike arrival times: A highly efficient coding scheme for neural networks ». In: *Parallel processing in neural systems* (Jan. 1990) (cit. on p. 6).

[126] [Video]. *Lab Scene Optical Flow Result*. 2017. URL: https://youtu.be/tTk1GbV7-9w (visited on 09/20/2017) (cit. on p. 56).

[127] [Video]. *Pipe Optical Flow Result*. 2017. URL: https://youtu.be/cjtPA0-tAEE (visited on 09/20/2017) (cit. on pp. 56, 57).

[128] [Video]. *Spiral Optical Flow Result*. 2017. URL: https://youtu.be/Nt9tlMyv2_Q (visited on 09/20/2017) (cit. on pp. 55, 56).

[129] N. X. Vinh, J. Epps, and J. Bailey. « Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance ». In: *The Journal of Machine Learning Research* 11 (2010), pp. 2837–2854 (cit. on p. 35).

[130] D. Vishwanath and E. Blaser. « Retinal blur and the perception of egocentric distance ». In: vol. 10. 26. 2010, pp. 1–16 (cit. on p. 5).

[131] B. Wandell, A. E. Gamal, and B. Girod. « Common principles of image acquisition systems and biological Vision ». In: vol. 90. 1. 2002, pp. 5–17 (cit. on p. 7).

[132] M. Watanabe and S. Nayar. « Rational filters for passive depth from defocus ». In: vol. 27. 1. 1997, pp. 203–225 (cit. on p. 7).

[133] A. B. Watson and A. J. Ahumada Jr. « A look at motion in the frequency domain ». In: (1983) (cit. on p. 41).

[134] T. Werner, E. Vianello, O. Bichler, D. Garbin, D. Cattaert, B. Yvert, B. De Salvo, and L. Perniola. « Spiking Neural Networks based on OxRAM Synapses for Real-time Unsupervised Spike Sorting ». In: *Frontiers in Neuroscience* 10 (2016), p. 474 (cit. on p. 40).

[135] N. H. Weste and D. Harris. *CMOS VLSI design: a circuits and systems perspective*. Pearson Education India, 2015 (cit. on p. 2).

[136] J. H. Wijekoon and P. Dudek. « Integrated circuit implementation of a cortical neuron ». In: *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*. IEEE. 2008, pp. 1784–1787 (cit. on p. 72).

[137] J. Wild, Z. Prekopcsak, T. Sieger, D. Novak, and R. Jech. « Performance comparison of extracellular spike sorting algorithms for single-channel recordings ». In: *Journal of neuroscience methods* 203.2 (2012), pp. 369–376 (cit. on pp. 31, 35, 36).

[138] B Wodlinger, J. Downey, E. Tyler-Kabara, A. Schwartz, M. Boninger, and J. Collinger. « Ten-dimensional anthropomorphic arm control in a human brain- machine interface: difficulties, solutions, and limitations ». In: *Journal of neural engineering* 12.1 (2014), p. 016011 (cit. on pp. 40, 73).

[139] A. Yousefzadeh, M. Jablonski, T. Iakymchuk, A. Linares-Barranco, A. Rosado, L. A. Plana, S. Temple, T. Serrano-Gotarredona, S. B. Furber, and B. Linares-Barranco. « On multiple AER handshaking channels over high-speed bit-serial bidirectional LVDS links with flow-control and clock-correction on commercial FPGAs for scalable neuromorphic systems ». In: *IEEE transactions on biomedical circuits and systems* 11.5 (2017), pp. 1133–1147 (cit. on p. 65).

[140] A. Yousefzadeh, L. A. Plana, S. Temple, M. T. Serrano Gotarredona, S. B. Furber, and B. Linares Barranco. « Fast predictive handshaking in synchronous FPGAs for fully asynchronous multisymbol chip links: Application to SpiNNaker 2-of-7 links ». In: *IEEE Transactions on Circuits and Systems-II-Express Briefs, 63 (8), 763-767.* (2016) (cit. on p. 65).

[141] C. Zhou, S. Lin, and S. Nayar. « Coded aperture pairs for depth from defocus and defocus blurring ». In: vol. 93. 1. 2011, pp. 53–69 (cit. on p. 7).