Feature Extraction and Pattern Recognition with Neuromorphic Vision Sensors

Jean-Matthieu Maro

Sorbonne Université

École Doctorale
Sciences mécaniques, acoustique, électronique et robotique de Paris
ED 391 - SMAER

Institut de la Vision, Équipe Vision et Calcul Naturel

Feature Extraction and Pattern Recognition with Neuromorphic Vision Sensors

par

Jean-Matthieu Maro

Thèse de Doctorat de Robotique

sous la direction de Ryad Benosman et Sio-Hoï Ieng

Présentée et soutenue publiquement le 30 septembre 2020

Devant un jury composé de :

Prof.	Elisabetta Chicca	Rapporteur
CR	Timothée Masquelier	Rapporteur
Prof.	Bruno Gas	Examinateur
Prof.	Ryad Benosman	Directeur de thèse
MC	Sio-Hoï Ieng	Directeur de thèse

RÉSUMÉ

Les caméras événementielles ou neuromorphiques sont des capteurs s'inspirant de l'oeil humain dans leur fonctionnement. Chaque pixel est indépendant, asynchrone et réagit aux changements de luminosité se produisant dans son propre champ de vision ; ces changements se traduisent par l'émission d'un événement. Le flot d'évènements émis par la caméra est une représentation éparse et non redondante de la scène visuelle. Dans cette thèse, nous introduisons des méthodes événementielles, c'est-à-dire asynchrones et déclenchées par des événements, qui ont pour but l'extraction de caractéristiques et la reconnaissance de formes. Nous introduisons une première primitive basée sur le flot optique. Celle-ci est ensuite utilisée pour des tâches de reconnaissance de gestes et de détection de coins. Nous introduisons ensuite un système complet de reconnaissance de gestes basé sur HOTS (Hierarchy Of Time-Surfaces, Lagorce 2016) avec suppression dynamique de l'arrière-plan. L'ensemble se contente de la puissance de calcul offerte par un smartphone standard. Nous présentons ensuite une proposition d'architecture de mémoire adaptée aux algorithmes événementiels semi-accumulatifs. Celle-ci se fonde sur le principe des mémoires associatives. Elle présente deux avantages : un stockage plus efficace des événements et la possibilité d'effectuer des calculs à l'intérieur même du bloc mémoire. Ce dernier point est un pas supplémentaire vers un calcul distribué dans les systèmes événementiels. Nous discutons également au long de ce travail des bénéfices et désavantages des méthodes semi-accumulatives.

ABSTRACT

Event-based cameras – also known as neuromorphic – are sensors that mimic the mammalian eye in the way they acquire the visual scene. Each pixel is independent and asynchronous, and reacts to changes in its own field-of-view by emitting events that signal these changes. The stream of events output by the camera is a sparse and non-redundant representation of the visual scene. In this thesis, we introduce event-based – asynchronous, triggered by new events – methods for feature extraction and pattern recognition. The first feature is a motion-based feature that accumulate optical-flows, that can serve for pattern recognition and corner detection. We then introduce a gesture recognition pipeline that is embarked on a smartphone, that includes a dynamic background suppression, and runs without any off-board requirements. We then present a new time-adaptive memory architecture for event-based algorithms, focusing on semi-accumulative methods. This memory is based on an associative memory, and while allowing for a more efficient storage of events, enables "in-memory" computation, pushing forward the philosophy of distributed computation in event-based systems. We finally discuss the benefits and disadvantages of semi-accumulative methods.

ACKNOWLEDGEMENTS

Maman, merci. *Je suis lancé.*

Papa, merci pour ton aide précieuse qui veut dire beaucoup.

Valbi, merci pour ce fameux traité.

Je veux également remercier ceux avec qui j'ai travaillé et ceux qui travaillaient à mes côtés : Laurent Dardelet, pour avoir partagé son second bureau avec moi. Quentin Sabatier & Laure Caruso, qui ont été là. Gregor Lenz, pour sa grenouille, et Christopher Reeves, pour son aide dans l'acquisition de NavGesture : sans vous deux, pas de démonstrateur à FG'2019 ! Ricardo Tapiador-Morales, pour nos collaborations fructueuses ! Xavier Clady, pour mes premiers pas dans l'*event-based*. Anne Marie, pour ta gentillesse. Merci ! Merci à tous les partenaires du projet ECOMODE, particulièrement Chris (encore lui !) du Streetlab, Nadia de FBK et l'équipe d'Experis : Antonio, Andrew et Gema. Les doctorants et post-docs du labo, qui m'ont beaucoup apporté par les discussions et l'entraide, et surtout pour les bons moments partagés : ils sont nombreux et se reconnaîtront.

Merci à *Ta Soeur* pour avoir payé leurs impôts année après année.

I would also like to thank my supervisors for giving me the opportunity to work in this field.

CONTENTS

1 INTRODUCTION

Despite the tremendous progress of Computer Vision during the last decades, efficient automated visual understanding of dynamic scenes still remains a challenge. This contrasts with the available interpretation of static images that has recently reached numerous milestones, thanks to the invention of GPU and their wide use in modern hardware and consumer market devices. It has now become "normal" to see every smartphone commonly equipped with built-in features such as face detection and many other advanced functionalities such as automated tagging of pictures. Until few years ago, such functionalities were only available in research laboratories and required powerful desktop computers to operate.

Unfortunately (or fortunately to us), all these developments could not be extended to efficiently handle dynamic visual scene processing that remains the next challenge of machine vision.

Visual representations of the real world have always been based on bidimensional static images, a relic from painting and early photography. Indeed, the first device designed to capture visual scenes was the *camera obscura*, a mechanism shown in Figure.1. This invention can be seen as the "mother" of all cameras. Its evolution over time followed the technologies available at each century: hand drawn canvas, photographic plates, then paper followed by celluloid film and more recently photo-diodes and pixels. Another linked extension has been the invention of motion pictures that relies on a "rapid" acquisition and display of a set of static pictures, leading to the invention of the cinematograph and the *motion-picture* industry, the television... Modern *frame-based* cameras widely used in computer vision are a relic of this acquisition principle and historic stream.

Figure 1: The camera obscura principle, also known as the pinhole camera. The visual scene is projected in a reversed and inverted manner on the wall opposing the hole. Illustration from James Ayscough's A short account of the eye and nature of vision (1755, 4th edition, public domain)

This quick historical perspective emphasizes the point that the dominant acquisition method [1] for dynamic visual scenes was originally designed

to acquire and display static scenes. The adaptation of this acquisition principle to dynamic scenes (initiated by Muybridge and Stanford, to solve the question about whether a galloping horse had always at least a foot on the ground) consisted in understanding that such scenes could be represented as a sequence of static subsequent images. By displaying such a sequence at a high enough framerate, one could obtain an illusion of fluent movement.

This invention although sufficient for display purposes, is unfortunately not the most efficient when applied to modern technologies. Its acquisition relies on updating the whole pixel array at a predefined rate, regardless to the scene's dynamic content. This has the vicious tendency to introduce both *under-* and *over*-sampling problems into the same acquisition. These two issues usually occur simultaneously, as the dynamical content of parts of the visual scene are usually different, as illustrated in Fig. 2-A. If the framerate is too low in regards to the dynamics of the scene, it will likely result in motion-blur or missed information as shown in Fig. 2-B. If the framerate is high enough, it will nonetheless results in over-sampling of the static background. Frames are dense representation, unnecessary information will then have to be wastefully processed at each frame. It is then very unlikely that such an inadequate acquisition method can lead to efficient algorithms as by construction it has been designed initially to acquire one static scene for display purposes.

Nevertheless, before the advent of neuromorphic engineering, the use of frames dominated (and still does) modern Computer Vision. As cameras' framerate and resolution were increasing over the years, more and more processing power, bandwidth for transmission and memory storage were required. To make things even worst, new techniques emerged, such as Convolutional Neural Networks (CNNs), requiring even more resources-hungry hardware such as Graphics Processing Unit (GPUs).

On the opposite side of the power requirement spectrum, the human brain outperforms existing modern computers in a variety of tasks such as general AI, visual scene understanding, language processing and many others. When considering the power consumption of brains of around 20-40 watts compared to the thousands of Watts required by modern computers, the difference is even more striking. The low energy requirements and functionalities of biological systems remain unmatched by artificial processing systems.

Neuromorphic – bio-inspired – sensing and processing, being also the scope of this Ph.D., aims at reproducing insights gained from neurobiological systems into artificial ones. Neuromorphic event-based vision sensors are part of an effort to provide more advanced, biologically plausible, functionalities to artificial systems, such as a more efficient visual processing and understanding of dynamic scenes. Unlike conventional cameras, these sensors perform local updates using a relative change acquisition paradigm that auto-adapts to the scene's dynamic contents (Fig. 3-A, 3-B). Furthermore, this acquisition method enforces that changes in the visual scenes are acquired only once at an adaptive frequency. This novel approach to visual acquisition requires to rethink computer vision algorithms to comply with the sparseness, the high temporal precision and the low latency of these sensors.

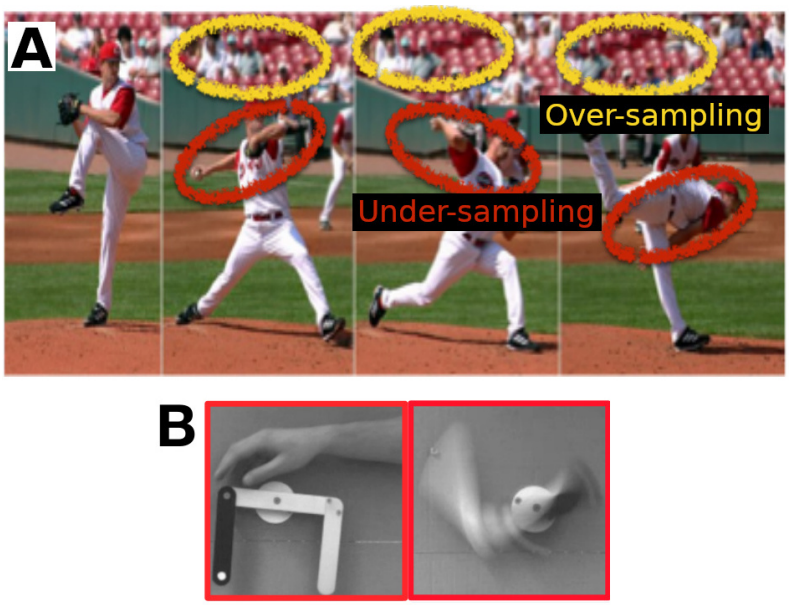


Figure 2: (A) Images from a conventional camera usually contain over- and under-sampled areas in the same frame because of the synchronous and full frame acquisition principle. If increasing the framerate allows for a clearer acquisition of the throw, it also results in increasing redundant information from the static background. (B) A pendulum is observed by a conventional camera (right), but the acquisition principle leads to motion-blur (left).

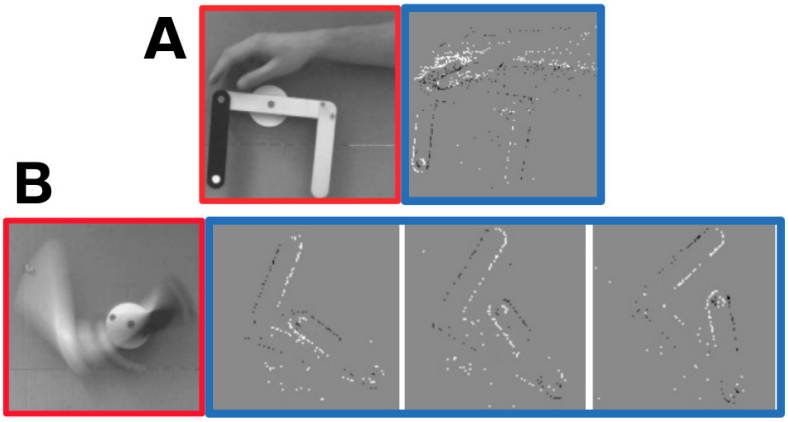


Figure 3: (A) The pendulum is observed by a conventional camera (red) and an event-based camera (blue). (B) in red the image from the conventional sensor shows motion blur due to the high velocity movement, while in (blue) there is no such effect (for display purposes we generated frames from events, as event-based cameras do not output images). Also, while the conventional camera is busy integrating pixels and transmitting the whole frame, the event based sensor allows a "continuous" acquisition of the dynamics of the pendulum's motion in the meantime.

EVENT-BASED CAMERAS

Event-based cameras differ from conventional cameras because each pixel is a relative change detector that operates individually and asynchronously, detecting luminance changes in its own field-of-view [2, 3, 4]. Pixels emit

events that signal these changes as soon as a relative luminance threshold is crossed as illustrated in Fig. 4-A, 4-B. The output of a simulated color version of such an event-based camera is represented in Fig. 4-C, 4-D). In these examples it becomes clear that using this acquisition paradigm, only changes are transmitted, and static backgrounds are not over-sampled. This results in the suppression of spatial redundancy, and a sparse and asynchronous encoding of the visual scene. The particular sensor shown in Fig. 4-B features an individual exposure measurement circuit in each pixel. Triggered by its associated change detector, this circuit encodes the absolute luminance as an inter-event interval [3]. However, in all this work we will only rely on events output by the change detector, never taking into consideration absolute luminance measurement events.

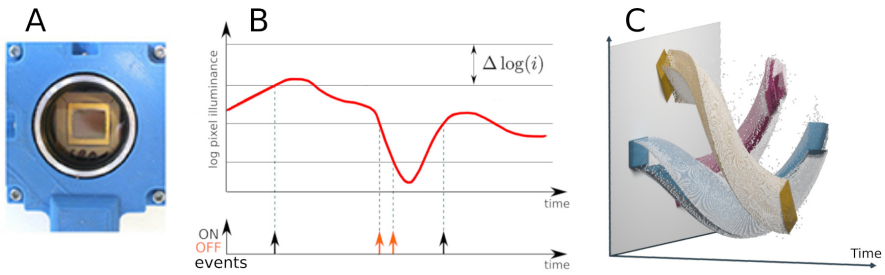


Figure 4: Event-based sensors operation principles: (A) The ATIS sensors used in this work [3]. (B) When a given pixel's luminosity change reaches a given threshold, it produces a visual event with an x,y address, a timestamp and a polarity, which is either ON or OFF depending on the relative luminosity change. (C) shows the stream of events generated by a simulation of three rotating colored shapes [5].

Formally, an event e can be defined as a triplet: $e = (\mathbf{x}, t, p)$, where $\mathbf{x}$ represents its spatial position in the sensor array, t its timestamp and p its polarity. This representation is referred as Address Event Representation (AER). For events output by the camera, the polarity encodes whether the detected change in luminance is positive or negative. But events can also be used as a mean to carry information as they start flowing into algorithms, and the polarity can then be used to encode other properties.

PROCESSING EVENTS

This new way of acquiring visual scenes has lead to several new options to process the incoming data. If frames are dense and global representations of the visual scene that are generally processed synchronously, the stream of events is on the contrary a sparse representation where events signal local changes, and these individual events can be processed asynchronously as they happen. This results in several dichotomies to be considered when proposing a nomenclature for algorithms that operate on event-based data:

- dense *vs.* sparse
- synchronous *vs.* asynchronous
- global *vs.* local

These dichotomies lead to a wide spectrum of algorithms, from synchronous and global frame-based techniques to asynchronous event-triggered and event-wise algorithms. An attempt of a nomenclature inspired from [6] is to consider 3 major trends in event-based data processing:

- *accumulative* methods are techniques that aim at creating a frame from received events, in order to take advantage of conventional computer vision techniques. They are synchronous and process dense representations;
- *non-accumulative* methods are asynchronous algorithms, triggered by each new incoming event, in order to update one or several internal models incrementally. Only the current event is considered by the algorithm, never taking into account older events: in these algorithms, events are processed only once, as soon as they are received, never to be considered again;
- *semi-accumulative* methods are also asynchronous and triggered by incoming events. However, they allow to consider spatio-temporal neighborhoods of events, instead of only one event. These methods are used in tasks ranging from local descriptors like time-surfaces [7] to local optical-flow computation [8]. They lie in-between accumulative and non-accumulative techniques and are prone to partial- and over-processing, that we further elaborate in the dedicated section.

Accumulative methods: the use of frames

Full-frame image and video reconstruction from events has quickly become an active trend in event-based vision [9, 10, 11, 12] in order to apply frame-based techniques such as CNNs [13, 14], optical flow computation [15, 16], object recognition [17, 18, 19, 20, 14] or image denoising [21].

However, if converting events into frames is an easy way to take advantage of five decades of research in frame-based computer vision, it completely neglects the fundamental shift in the acquisition principle [22] between conventional cameras and neuromorphic sensors, that requires to rethink how to process visual data [23]. Using frame-based algorithms on event-based data can hardly result into efficient processing and misses out on the properties of sparse event-based representations:

- the sparse stream of events is replaced by a dense representation, leading to additional computation as the whole frame must be processed;
- increased memory consumption footprint because of the dense representation. Data transmission of the whole frame at once can also be an issue;
- the precise timing of events is *usually* lost in the process;
- latency is likely to increase, because frames are *usually* integrated over time, and can require more processing time than single or small groups of events that can be processed as soon as they are received.

Non-accumulative methods: pure event-wise computation

The principle is to perform a quick calculation, triggered at each new incoming event [24, 25], in order to update a model. These methods make the assumption that each single event carries a small bit of information that can be used to incrementally update a model or a representation. The "blob" tracker [26] is a simple example: event-wise computations are used to update the characteristics of the distribution that defines the tracker. The Dynamic Background Suppressor presented in this work is another illustration of such an algorithm, in which each incoming event updates an activity score that is used to discriminate the background clutter from the useful foreground signal.

Besides reducing latency to a minimum, these methods have major advantages: as each event is processed only once, these algorithms make sure that no redundancy is introduced during the processing phase. However, it seems that certain algorithms can not be written in a non-accumulative form, especially when local spatio-temporal neighborhoods are required.

Semi-accumulative methods: the best of both worlds?

Semi-accumulative methods are asynchronously triggered algorithms, like the non-accumulative ones. However, instead of considering only the incoming event, these methods make also use of older events in the spatio-temporal neighbor of this event. Semi-accumulative methods are used in numerous tasks that require to consider local contexts such as optical-flow or feature extraction [7, 8, 27, 28, 29, 30, 31]. Behind these methods lay the idea that the whole is greater than the sum of its parts, and that events grouped into a spatio-temporal neighborhood can yield more information than it is possible to obtain using the incremental updates of non-accumulative methods. Hence, semi-accumulative methods lie in-between the two previously presented techniques. They allow for small local computations because triggered by new incoming events. If well-designed, they preserve most of the sparseness of the data and avoid dense representations, meaning that computation happens *when* and *where* a change is detected in the visual scene. This contrast with frame-based algorithms that must process the complete frame in order to detect changes in the visual scene.

However, one must be careful when designing semi-accumulative algorithms as older events can be processed several times, as newer events happen in their spatial neighborhood. Due to the fact that events are received and processed in a sequential order, it can lead to both *over*-processing, and its antagonist problem, *partial*-processing:

- at first, when a neighborhood contains few events, there is the risk to process partial incomplete information in regard to the stimulus that is being acquired;

- then, as events accumulate in the neighborhood, there is the risk to process older events again and again, leading to over-processing of redundant information.

It must be noted that partial- and over-processing can not happen in non-accumulative algorithms as events are processed only once and then forgotten. This could also be enforced for accumulative algorithms if the binning of events is non-overlapping from frame to frame.

A first attempt to address this partial- and over-processing issue in semi-accumulative algorithms is proposed in Chapter 3 as a simple heuristic that prevents some of the spatio-temporal neighborhoods of incoming events from being processed.

However it must be noted that asynchronous processing of event-based data using semi-accumulative methods still lacks a theoretical background or an empirical framework, and should in the meanwhile be designed taking this issue into account, as it can lead to unexpected partial- and over-processing.

PH.D. WORK

As introduced previously, event-based sensing and processing define a new promising paradigm that does not resemble the conventional one relying on frames and that leads inevitably to poorly managed resources. The objective of this thesis is to tackle the problem of developing efficient on-the-edge methodology and processing methods, in the form of non-accumulative or semi-accumulative algorithms.

This thesis also aims to contribute to the state-of-the-art of machine learning by developing alternative techniques that deal directly with the bottlenecks of modern machine learning: the requirements of large amounts of data for training, and the processing of dense frames. If events signal local updates in the visual scene, the *absence* of events signals that no processing is needed.

This work starts with the development of a motion-based feature that makes use of local optical-flow (Article 1). Then we proceed with the use and adaptation of a low level feature and event based machine learning algorithm using time-surfaces introduced in [7] (Article 2). This work on low level information extraction is the first building block required by the higher objective of solving a gesture recognition task for mobile phone applications.

This Ph.D. work was partially funded by the European project of ECOMODE (Event-Driven Compressive Vision for Multimodal Interaction with Mobile Devices), that defined a constrained framework that aims at using the resources of a mobile phone connected to an event-based camera. The aim of ECOMODE is to facilitate the use of modern smartphones and tablets by the elderly and the visually-impaired, by enabling their operation using air gestures through an event-based camera. Since the gesture recognition pipeline must run in real time on a smartphone, it imposes several constraints, such as memory consumption and available processing power. This is an ideal test-bed for event-based sensing and processing. Furthermore, considering more practical aspects, several methods had to be developed for non-controlled environments. Their scope is to enable to cope with background clutter or jerky movements as smartphones are used on the go. We

also had to develop new datasets that would match as closely as possible real world use-cases.

The tight power, memory and processing budget imposed by the smartphone required the optimization of event processing. This opened a deeper thought on the semi-accumulative approach that we chose to use. The gesture recognition module has been ported on FPGA to study the power consumption of a future chip and provide a comparison with existing state-of-the-art implementation on neuromorphic chips such as TrueNorth [13] (Article 3).

The insight gained in the course of ECOMODE also revealed that current computers and software architectures are not designed for efficient event-based computation and storage. Storing events in image-like representations proved itself inefficient when applied to the sparse properties of event-based cameras, often requiring much more memory allocation than needed when considering only the sufficient events to perform gesture recognition. Semi-accumulative methods such as the event-based mechanism recommends, require the retrieval of a spatio-temporal neighborhood around each incoming event. This further emphasizes the inadequacy of standard memory architecture. To improve the processing of events, we introduce a new hardware alternative, based on in-memory computation using associative memories. These reduce considerably the context retrieval around incoming events (Article 4). This approach is a new step towards distributed, parallel computation instead of relying on CPU-centred computation.

OUTLINE OF THIS WORK

This document is organized as follows:

- Chapter 2:

  A Motion-Based Feature (MBF) is presented. The MBF is a self-decaying accumulation of the discretization of the optical-flow, velocity-wise and direction-wise. Although the MBF is based on dynamic properties, it allows for the detection of static properties, such as corners. This feature is used in a first gesture recognition pipeline. Results on a compact preliminary dataset are presented;

- Chapter 3:

  This chapter introduces a gesture recognition pipeline that operates in real time on the ECOMODE smartphone prototype. We adapted the learning mechanism and time-surface computation introduced in [7] to better match available and limited mobile phones computation capabilities. We also introduce a non-accumulative background suppression as part of the pipeline. Two novel datasets are introduced. The chapter also provides accuracy results on the widely used DvsGesture dataset;

- Chapter 4:

  In this chapter, we introduce the time adaptive resource architecture, suited for semi-accumulative methods, with the objective to push further the optimization and efficiency of the gesture pipeline, and more

generally event-based processing. We analyze several widely-used datasets to assess the reduction in memory footprint. We then show that by using time adaptive Content-Addressable Memories, it is possible to perform "in-memory" computation in order to compute time-surfaces and HOTS prototype matching.

- Chapter 5:

  Finally, this chapter concludes this work based on the main contributions of this thesis.

LIST OF PUBLICATIONS

1. Clady, X., Maro, J. M., Barré, S., & Benosman, R. B. (2017). A motion-based feature for event-based pattern recognition. Frontiers in neuroscience, 10, 594.

2. Maro, J. M., Ieng, S. H., & Benosman, R. (2020). Event-based gesture recognition with dynamic background suppression using smartphone computational capabilities. Frontiers in Neuroscience, 14, 275.

3. Tapiador-Morales, R., Maro, J. M., Jimenez-Fernandez, A., Jimenez-Moreno, G., Benosman, R., & Linares-Barranco, A. (2020). Event-Based Gesture Recognition through a Hierarchy of Time-Surfaces for FPGA. Sensors, 20(12), 3404.

4. Maro, J. M., Tapiador-Morales, R., Ieng, S. H., Linares-Barranco, A. & Benosman, R. (2020) Why Generating Frames from Neuromorphic Event-based Cameras is Wrong: The Need of Incremental Computation and Time Adaptive Resources Management. **(In Preparation)**

5. Mana, N, Reeves, C, Lenz, G, Maro, J.M., Linares-Barranco, B, Serrano-Gotarredona, T, Camuñas-Mesa, L, Bartolozzi, C & Benosman, R. ECO-MODE Review: Mid-air Gesture and Voice Interaction with Mobile Devices by Elderly and Visually Impaired Users. **(In Preparation)**

CONFERENCES

1. Maro, J. M., & Benosman R. (2017). Mid-air Gesture Recognition Using an Event-based Vision Sensor. Workshop on Designing, Implementing and Evaluating Mid-Air Gestures and Speech-Based Interaction. Computer-Human Interaction Italy (CHItaly 2017).

2. Maro, J. M., Lenz, G., Reeves, C., & Benosman, R. (2019, May). Event-based Visual Gesture Recognition with Background Suppression running on a smart-phone. In 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019) (pp. 1-1). IEEE.

3. Best Demo Award, 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019).

2 A MOTION-BASED FEATURE FOR EVENT-BASED PATTERN RECOGNITION

This chapter introduces an event-based luminance-free feature from the output of asynchronous event-based neuromorphic retinas. The feature consists in mapping the distribution of the optical flow along the contours of the moving objects in the visual scene into a matrix. Asynchronous event-based neuromorphic retinas are composed of autonomous pixels, each of them asynchronously generating "spiking" events that encode relative changes in pixels' illumination at high temporal resolutions. The optical flow is computed at each event, and is integrated locally or globally in a speed and direction coordinate frame based grid, using speed-tuned temporal kernels. The latter ensures that the resulting feature equitably represents the distribution of the normal motion along the current moving edges, whatever their respective dynamics. The usefulness and the generality of the proposed feature are demonstrated in pattern recognition applications: local corner detection and global gesture recognition.

2.1 INTRODUCTION

In computer vision, a feature is a more or less compact representation of visual information that is relevant to solve a task related to a given application (see [32, 33, 34, 35, 36, 37, 38]). Building a feature consists in encoding information contained in the visual scene (global approach) or in a neighborhood of a point (local approach). It can represent static information (e.g. shape of an object, contour, etc.), dynamic information (e.g. speed and direction at the point, dynamic deformations, etc.) or both simultaneously.

We propose a motion-based feature computed on visual information provided by asynchronous image sensors known as neuromorphic retinas (see [39, 40]). These cameras provide visual information as asynchronous event-based streams while conventional cameras output it as synchronous frame-based streams. The ATIS ("Asynchronous Time-based Image Sensor", [41, 40]), one of the neuromorphic visual sensors used in this work, is a time-domain encoding image sensor with QVGA resolution. It contains an array of fully autonomous pixels that combine an illuminance change detector circuit, associated to the PD1 photodiode, see Fig. 5.a) and a conditional exposure measurement block, associated to the PD2 photodiode. The change detector individually and asynchronously initiates the measurement of an exposure/gray scale value only if a brightness change of a certain magnitude has been detected in the field-of-view of the respective pixel, as shown in the functional diagram of the ATIS pixel in Fig. 5.b and in Fig. 6. The exposure measurement circuit encodes the absolute instantaneous pixel illu-

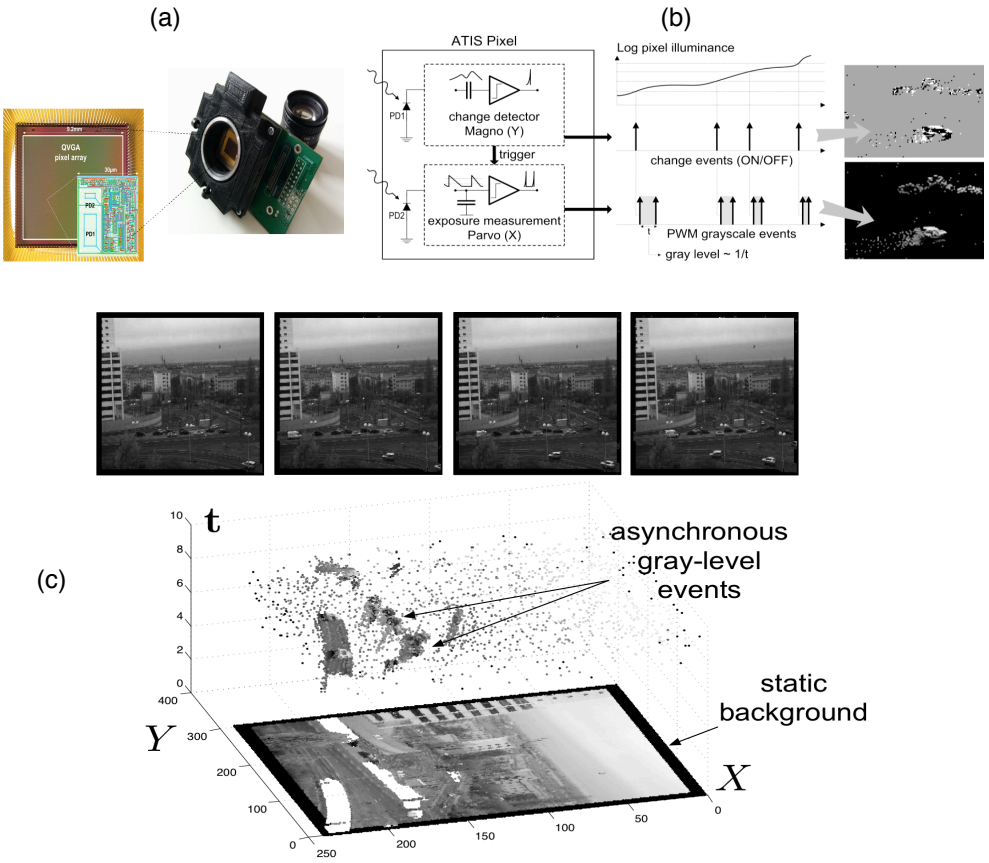


Figure 5: ATIS, Asynchronous Time-based Image Sensor: (a) The ATIS and its pixel array, made of 304x240 pixels (QVGA). PD1 is the change detector, PD2 is the grayscale measurement unit. (b) When a contrast change occurs in the visual scene, the ATIS outputs a change event (ON or OFF) and a grayscale event. (c) The spatio-temporal space of imaging events: static objects and scene background are acquired first. Then, dynamic objects trigger pixel-individual, asynchronous gray-level events after each change. Frames are absent from this acquisition process. Samples of generated images from the presented spatio-temporal space are shown in the upper part of the figure.

minance into the timing of asynchronous event pulses, more precisely into inter-event intervals. The DVS ("Dynamic Visual Sensor", [42, 43]), another neuromorphic camera used in this work, works in a similar manner but only the illuminance change detector is implemented and retina's spatial resolution is limited to 128×128 pixels.

Despite the recent introduction of neuromorphic cameras, numerous applications have already emerged in robotics (see [45, 46, 47, 48, 49, 50]), shape tracking (see [51, 52, 24]), stereovision (cf. [53, 54, 55, 56]), corner detection ([57]) or shape recognition (see [58, 59, 60, 61, 62]). This strong interest in such a sensor is essentially due to its ability to provide visual information as a high temporal resolution, luminance-free and non-redundant stream. This makes it a fitting for high-speed applications (e.g. gesture recognition as in [63], high-speed object tracking as in [64, 65]).

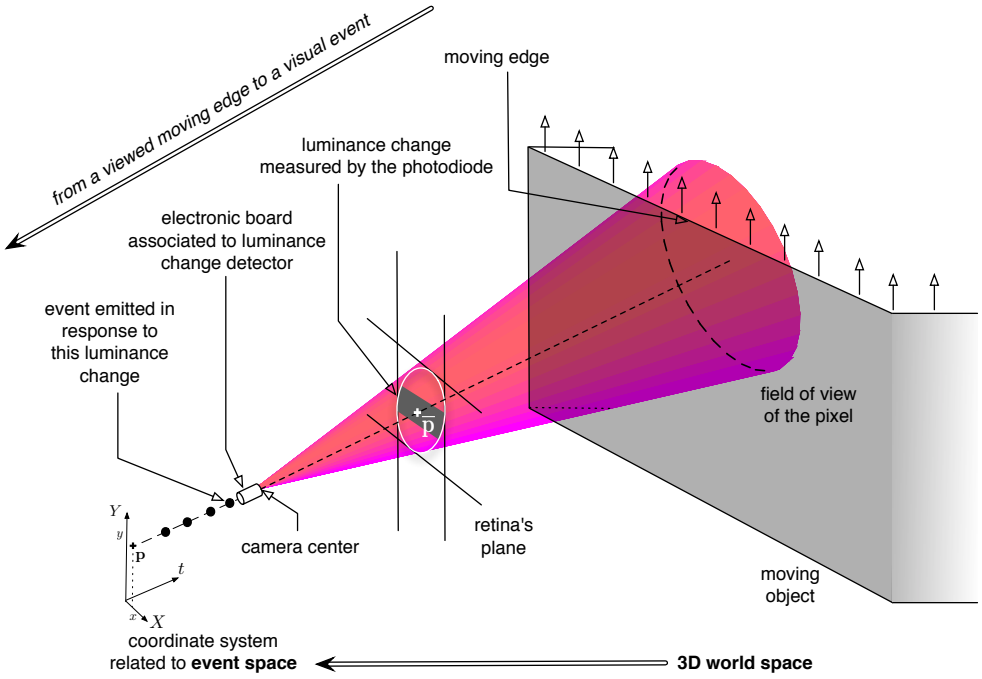


Figure 6: Illustration of the luminance change measured by a neuromorphic pixel, modeled as a cone-pixel ([44]), viewing an moving edge. $\mathbf{p}$ are the coordinates of the center of the pixel, in the coordinate system related to the retina's plane. The emitted event in response to this luminance change is represented as a black dot in the coordinate system XYt related to the event space (in the lower-left part of the figure).

The proposed feature consists in mapping the distribution of the optical flow along the contours of the objects in the visual scene into a matrix (see Section 2.2). It can be computed locally or more globally according to the targeted applications. Indeed, in the experimental evaluations, we propose to demonstrate its usefulness and generality in various applications. It is used to locally detect corners (see Section 2.3) or to summarize global motion observed in a scene in order to recognize actions, here hand gestures for an application in human-machine interaction (see Section 2.4).

2.2 MOTION-BASED FEATURE

Visual event streams are generated asynchronously at a high temporal resolution, essentially by moving edges. They are thus especially suitable for visual motion flow or optical flow (OF) computation ([66, 67, 68]) along contours of objects. In the following sections, methods and mechanisms are proposed to estimate normal motion flows computed around events and to map them into a matrix in order to incrementally estimate scene motion distribution (locally or globally). This matrix will be considered as a feature. Its computation requires only the visual events provided by the change detectors of the retina (associated to photodiodes PD1 in Fig. 5.a), that can be

defined as four components vectors:

$$\mathbf{e} = (\mathbf{p}, t, pol)^T, \tag{1}$$

where $\mathbf{p} = (x, y)^T$ is the spatial coordinate of each event, t, its timestamp and $pol \in \{-1, 1\}$ is the polarity, which is equal to -1/1 when the measured luminance decrease/increase is significant enough (see upper part of Fig. 5.b).

2.2.1 Extracting normal visual motion

We use the event-based OF computation method proposed in [66] which is known for its robustness and its algorithmic efficiency (see [48, 57, 69]). More bio-inspired event-based OF computation methods such as [68] and [67] can be used but they are computationally more expensive.

A function Σ_e that maps to each $\mathbf{p}$ the time t is defined locally:

$$\Sigma_e : \begin{array}{c} \mathcal{N}^2 \to \mathcal{R} \\ \mathbf{p} \mapsto t \end{array}$$

Applying the inverse function theorem of calculus, the vector $\nabla\Sigma_e$ measures the rate and the direction of change of time with respect to space: it is the normal optical flow, noted $\mathbf{v} = (v_x, v_y)^T$, such as:

$$\nabla\Sigma_e = \left(\frac{1}{v_x}, \frac{1}{v_y}\right)^\top$$

This equation could be defined assuming that the surface described by the visual events (generated by a moving edge) in the space-time reference frame $(XYt)^T$ is continuous. This assumption is validated through a regularization process proposed in order to locally estimate this surface as a spatiotemporal plane (fitted directly on the local event stream). In this work the implementation proposed in [57] has been chosen because it proposes mechanisms to automatically adapt the temporal dimension of the local neighborhood to the edge's dynamics, and to reject estimations of optical flow probably wrong and due to noise. This algorithm allows us to consider a function that associates for each valid visual event $\mathbf{e} \in \mathcal{E}$, a so-called visual motion event, noted $\mathbf{v_e}$, such as:

$$\begin{array}{c} \mathcal{E} \to \mathcal{V} \\ \mathbf{e} = (\mathbf{p}, t, pol)^T \mapsto \mathbf{v_e} = (\mathbf{p}, t, v, \theta)^T \end{array} \tag{2}$$

where $(v, \theta)^T$ corresponds to the intensity (i.e. speed) and the direction of the normal visual flow.

Remark 1 *Note that the polarity of visual events is not conserved by the function (Eq. 2). Indeed, in the applications proposed in this article, it is not useful to "memorize" if the visual flow has been computed on a positive or negative event stream. If required, the feature can be augmented in order to distinguish the distribution along "positive contours" from the one along "negative contours".*

2.2.2 Computing and updating the feature

As we said, the feature corresponds to the estimated distribution of the optical flow along the (local or global) contours in the visual scene. This distribution is evaluated on a grid-based sampling in the polar reference frame of the visual flow, such as it is subdivided into an interval set $\left\{\overline{\mathbf{v}^l}\right\}_l = \left\{(\overline{\theta^l}, \overline{v^l})^T\right\}_l$ where $\overline{\theta^l}$ is an angle based interval and $\overline{v^l}$ is an intensity based interval. Such a discretization of the velocity subspace is consistent with biologic observations about orientation (cf. [70, 71]) and speed (cf. [72]) selectivity in V1 cells and human psychophysical experiments about speed discrimination as in [73, 74, 75]. Here we parametrize the grid sampling mostly according to these biologic observations and human psychophysical experiments. However its ranges and precisions could be set in relation with targeted tasks, optimizing them according to given performance criteria. We define the centers $\{\theta^l\}_l$ of the angle intervals such as: $\theta^l \in [0, ..., 2\pi\frac{i}{N_\theta}, ..., 2\pi\frac{N_\theta-1}{N_\theta}]$, with $i \in [0, N_\theta - 1]$; $\frac{2\pi}{N_\theta}$ is the length of the interval and thus the angular precision of the grid. With $N_\theta = 36$, we barely reach the precision (approximatively 10°) observed for V1 simple cells (see [70, 71]). For the velocity intensity, we propose a non-regular speed-based sampling, where $\{v^l\}$ are the centers of the speed based intervals on a logarithmic scale. The sampling is then operated such that $v^l \in [v_{min}, ..., v_{min}\gamma^i, ..., v_{min}\gamma^{N_v-1}]$, with $i \in [0, N_v - 1]$ and $\gamma = 1 + \epsilon_v$ ($\epsilon_v > 0$). This discretization strategy ensures an *a priori* constant relative precision in speed estimation: $\frac{\Delta v^l}{v^l} \approx \epsilon_v$. Setting ϵ_v to 0.1 will barely correspond to the relative speed-discrimination threshold (10%) observed in human psychophysical experiments (see [73, 74, 75]). v_{min} has been fixed to $1 pixel.s^{-1}$ and N_v to 73 in order that $v_{max} = v_{min}\gamma^{N_v-1}$ is close to $1000 pixels.s^{-1}$, i.e. inversely close to the temporal precision of the visual events, estimated over 1ms (cf. [60]). Motions with intensities less than v_{min} are then discarded: they are assumed as belonging to static or faraway objects in the background visual scene. Motions with intensities higher than v_{max} are also discarded because noise associated to their computation can *a priori* be considered as too high.

Finally, the feature, noted $\mathbf{F} \in \mathcal{F}$, is defined as a matrix corresponding to this grid, and associated to a spatiotemporal point $(\mathbf{p}, t)^T$ of the retina (or to the entire visual scene for a global approach), and computed as:

$$\begin{array}{c} \mathcal{V} \to \mathcal{F} \\ \{\mathbf{v}_j\}_{j=1,...,N} \mapsto \mathbf{F}_{\mathbf{p},t}(v^l, \theta^l) = \sum_j w_v(v_j - v^l, \theta_j - \theta^l) w_s(\mathbf{p} - \mathbf{p}_j) w_t^l(t - t_j) \end{array} \tag{3}$$

where:

- w_t is a temporal exponentially decay function (or kernel), inspired by the synchrony measure of spike trains proposed in [76], such that:

$$w_t^l(t - t_j) = H(t - t_j) \exp\left(-\alpha v^l (t - t_j)\right) \tag{4}$$

  where $H(\cdot)$ is the Heaviside step function and α parametrizes the global decreasing dynamic. In our experiments (see Sections 2.3 and 2.4), we

fixed α to 0.8, i.e. close to 1 in order to mostly take into account the current edges while slightly smoothing them in order to make $\mathbf{F}$ less sensitive to both noise and missing data. This kernel gives indeed more weight (or a higher probability value) to events generated by current edges, i.e. the events with timings close to t, while also respecting an isoprobabilistic representation of the edges whatever their dynamics, as we will discuss below (see Section 2.2.3). Of course, other temporal kernels (Gaussian-based in [77],...) can be envisioned, but this one has the advantage of being causal and of leading to an incremental computation of the feature (see Eq. 7).

- w_s is a spatial bivariate function, which can be defined as:
  1. in a global approach, $w_s(\mathbf{p}-\mathbf{p}_j) = 1$, which gives an equitable representation to the edges whatever their spatial locations, or
  2. in a local approach:

$$w_s(\mathbf{p}-\mathbf{p}_j) = \frac{1}{2\pi\sigma_s^2}\exp\left(-\frac{\|\mathbf{p}-\mathbf{p}_j\|^2}{2\sigma_s^2}\right), \tag{5}$$

  where σ_s implicitly parametrizes the spatial scale of a region of interest or neighborhood around the spatial location $\mathbf{p}$; $\mathbf{F}_{\mathbf{p},t}$ then represents the local distribution of the normal velocities around the spatiotemporal location $(\mathbf{p}, t)^T$;

- w_v is the multiplication of two univariate Gaussian-like functions used to take into account potential imprecisions in the computation of the optical flow, defined as:

$$w_v(v_j - v^l, \theta_j - \theta^l) = \exp\left(-\frac{(v_j - v^l)^2}{V^2}\right)\exp\left(-\frac{(\theta_j - \theta^l)^2}{\Theta^2}\right) \tag{6}$$

  with $V^2 = v_j v^l$ in order to consider a relative speed imprecision, and Θ set to $20°$. So, even if an estimated motion belongs to a wrong interval because of noise, it will still contribute to the right element of the matrix, probably close.

As we said previously, the feature can be incrementally updated at each occurring visual motion event $\mathbf{v}_i$, considering that $\mathbf{F}_{\mathbf{p},0}(v^l, \theta^l) = \frac{1}{N_\theta N_v}$ for all $(v^l, \theta^l)^T$ (in order to consider, at time $t = 0$, an uniform distribution for the considered velocity-space), such as:

$$\mathbf{F}_{\mathbf{p},t_i}(v^l, \theta^l) = \mathbf{F}_{\mathbf{p},t_{i-1}}(v^l, \theta^l)\exp\left(-\alpha v^l(t_i - t_{i-1})\right) + w_s(\mathbf{p}-\mathbf{p_i})w_v(v_i - v^l, \theta_i - \theta^l) \tag{7}$$

Remark 2 *The feature works like a voting matrix, i.e. each visual motion event votes for the speed and direction interval it belongs (and its neighboring intervals through the weighting kernel w_v, Eq. 6). More visual events there are, more robust*

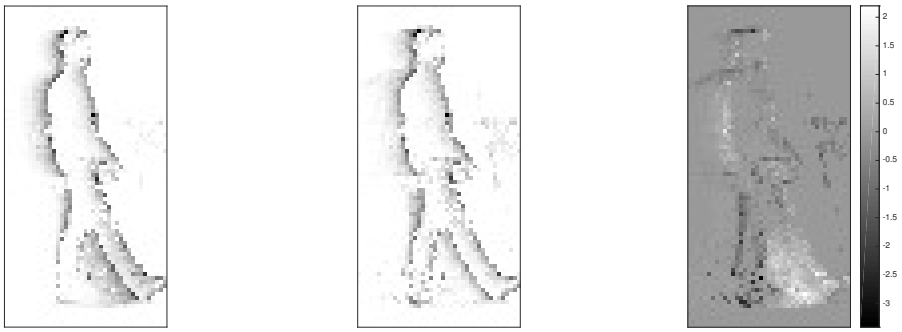
the feature will be. Conversely, the feature will be more sensitive to noise in low light or low contrast situations.

In addition the feature $\mathbf{F}$ *can be related to a probabilistic distribution while normalizing it to sum up to 1, i.e. to divide it with* $\sum_l \mathbf{F}(v^l, \theta^l)$.

In the global approach, $\mathbf{F}_{\mathbf{p},t}$ *is independent of* $\mathbf{p}$*; it can then be noted* $\mathbf{F}_t$*. Note that the feature is noted* $\mathbf{F}$ *(without sub-index) in this article when the application context (local or global approach) is not relevant or obvious.*

2.2.3 Speed-tuned *vs.* fixed decreasing strategies

Another important point to highlight is that the temporal decreasing function w_t (Eq. 4) is related to the speed v^l. Indeed, $\tau^l = \frac{1}{v^l}$ is the time during which an edge travels through a pixel, or, in other words, the estimated lifetime of its observation at a given location $\mathbf{p}$, as already remarked in [57, 69]. Including it as decay factor in the temporal kernel (Eq. 4 and 7) provides a more isoprobabilistic representation of the moving edges in $\mathbf{F}$, i.e. depending only of their contrasts whatever their respective dynamics.



(a) $I_{\tau=cst}$ with a fixed decreasing factor

(b) $I_{\tau=\frac{1}{v}}$ with a speed-tuned decreasing factor

(c) comparison $I_{\tau=cst} - I_{\tau=\frac{1}{v}}$

Figure 7: Illustration of different strategies for the exponential decay function; comparison between synchrony images I built applying exponential temporal kernels with a constant decreasing factor (a) and with a speed-tuned decreasing factor (b) to an event stream (acquired from a visual scene containing a walking person). The comparison $I_{\tau=cst} - I_{\tau=\frac{1}{v}}$ (c) of both images shows that the second strategy provides a more isoprobabilistic representation of the edges (taking into account the observation lifetime of the moving edges as in [57, 69]) than the first one; the high-velocity edges (resulting from the moving and forward leg) are over-represented and the low-velocity edges (resulting from the backward leg) are under-represented in the left synchrony image.

In order to concretely illustrate this point, Figure 7 represents two synchrony images I built integrating a visual event stream and with two different strategies for decay factor τ (related to the speed or not), such as, for each occurring visual event $\mathbf{e}_i$, $I(\mathbf{p}, t_i) = I(\mathbf{p}, t_{i-1}) \exp\left(-\frac{t_i - t_{i-1}}{\tau}\right) + \delta(\|\mathbf{p} - \mathbf{p_i}\|)$

where $\delta(\cdot)$ is the Dirac function. The left image (Fig. 7.a) results from this equation with a constant $\tau = cst$ (whatever the dynamics of the edges), and the middle image (Fig. 7.b) with a speed-tuned $\tau = \frac{1}{v}$. As shown in the right image (Fig. 7.c), which is the subtraction of both previous images without a speed-tuned factor the high-velocity edges (resulting from the moving and forward leg) are over-represented and the low-velocity edges (resulting from the backward leg) are under-represented in the corresponding synchrony image (Fig. 7.a). The moving edges are more equitably represented in the second synchrony image (Fig. 7.b) with a speed-tuned temporal kernel and, by extension, in feature $\mathbf{F}$. Results in Section 2.3.2 show this equitable representation is very important to obtain accurate results.

The proposed strategy is also consistent with biological observations. Indeed [78] showed that the effective integration time of the computations in direction-selective cells changes with stimulus speed; the integration time for slow motions is longer than that for fast motions. This is modeled in Eq. 4 as a decay factor inversely proportional to the speed intensity.

Algorithm 1 Computation of the local feature

1: **for all** pixel's location $\mathbf{p} \in$ Retina **do**
2: Set $\mathbf{F}_{\mathbf{p},0}(v^l, \theta^l) = \frac{1}{\mathbf{N}_\theta \mathbf{N}_v}$ for all $(v^l, \theta^l)^T$
3: **end for**
4: **for all** event $\mathbf{e} = (\mathbf{p}, t, pol)^T$ **do**
5: Compute the current optical flow $\mathbf{v_e} = (\mathbf{p}, t, v, \theta)^T$ (see Section 2.2.1).
6: **for all** $\mathbf{p}_i \in \Omega_\mathbf{p}$, where $\Omega_{\mathbf{p}_i}$ is a spatial neighborhood such as $\| \mathbf{p} - \mathbf{p}_i \| < 2\sigma_s$, **do**
7: Update $\mathbf{F}_{\mathbf{p}_i,t_i}$: $\mathbf{F}_{\mathbf{p}_i,t}(v^l, \theta^l) = \mathbf{F}_{\mathbf{p}_i,t_i}(v^l, \theta^l) \exp\left(-\alpha v^l (t - t_i)\right) + w_s(\mathbf{p} - \mathbf{p}_i) w_v(v - v^l, \theta - \theta^l)$, where t_i is the timing of the previous update of $\mathbf{F}_{\mathbf{p}_i,t}$ (see Eq. 7)
8: **end for**
9: **end for**
10: Output $\mathbf{F}_{\mathbf{p},t}$

The organization of the feature in a polar coordinate frame based grid, greatly facilitates its computation and its update. The representation of the visual motion information into speed and direction coordinates grants that each speed-tuned decay factor can be associated to an element of the grid, and not directly to the velocity associated to the occurring visual motion event. The latter indicates only which elements in the grid have to be incremented. A bio-inspired implementation can be envisioned where visual motion events are conveyed by selective lines (each line conveying only the motion events $\mathbf{v_e}$ included in its associated interval, $(v, \theta)^T \in (\overline{v^l}, \overline{\theta^l})^T$) from a neuron layer computing the optical flow to a leaky integrate-and-fire (LIF) neural layer (cf. [79]), in which each neuron could be assimilated with an element of the feature; this selectivity of lines could result from the selectivity of neurons in the first neuron layer.

Indeed the following model (notations are inspired by [62]) can be used to update the membrane potential of a LIF neuron for a given input event (or spike):

$$V_{mp}(t_i) = V_{mp}(t_{i-1}) \exp(-\frac{t_i - t_{i-1}}{\tau_{mp}}) + w_k w_{dyn} \quad (8)$$

where τ_{mp} is the membrane time constant, w_k is the synaptic weight of the k-th synapse (through which the input event or spike arrives) and w_{dyn} is a dynamic weight controlling a refractory period (see [79, 62] for more details). This model is very similar to the incremental updating equation of our feature, Eq. 7. The only things missing are the dynamic weight w_{dyn} and a firing threshold V_{th} in order to output approximatively the value of the corresponding feature's element as an event stream (or spike train), and then approximatively following a rate-coding model. Here, the refractory period should be set close to 0 (probably as a small fraction of the integration time $\tau^l = \frac{1}{v^l}$), in order to allow (quasi-)simultaneous visual events in the neighborhood (i.e. the events generate by the same contour moving across several pixels in the neighborhood) to contribute equitably to the neuron's potential, i.e. the value of the corresponding element of the feature.

For the local approach, a leaky integrate-and-fire neural layer has to be implemented for each pixel; this neural layer collects the visual motion events from the receptive field, Ω_{p_i} (defined as $\| \mathbf{p} - \mathbf{p}_i \| < 2\sigma_s$) defined by the corresponding bi-variate spatial kernel (Eq. 5). This local computation is detailed in Algorithm 1. For the global approach, only one neural layer is required, collecting the visual motion events estimated over the entire retina.

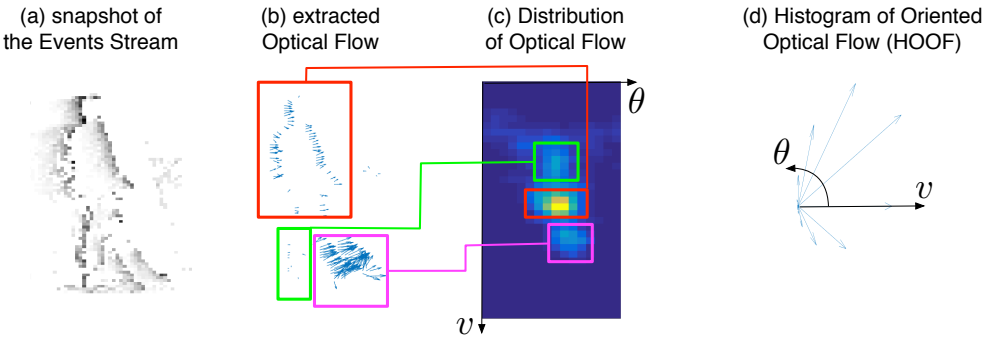


Figure 8: Illustration of the global motion-based feature for event-based vision: from the stream of events (a), the optical flow (b) is extracted. The feature corresponds to the distribution of this optical flow (c) in a polar coordinate frame, and can be reduced into a more compact and scale-invariant representation, called Histogram of Oriented Optical Flow (d) (see Section 2.4.1). As we can see in Figures (b) and (c), the motions generated by the forward leg (magenta boxes), the backward leg (green boxes) and the rest of body (red boxes) corresponds to three distinct and representative modes in the proposed feature.

Finally, Figure 8 shows that the distribution of optical flow representation in the global approach (Fig. 8.c) summarizes the principal motions observed in the visual scene. This property will allow us to propose a machine learning based approach to recognize gestures in Section 2.4. In the next Section, we will demonstrate that the local version can be also used to detect partic-

ular interest points, i.e. corners.

Remark 3 *If the photodiode of the retina's pixel is not square as for the ATIS's one (see [41] and Fig. 5.a), the frequency of a set of events emitted by a pixel will be not the same when a contour moves horizontally or vertically in the pixel's field of view (contour's speed and contrast are considered equal in both cases), because the contour travels the same surface of the photodiode during different time periods. In this case, keeping a decay factor invariant whatever the direction of the motion will introduce a bias, favoring one direction over another, in* $\mathbf{F}$*. To avoid this bias, a cone-pixel with an ellipse-based basis (and not a disk-based basis as illustrated in Figure 6) can be implicitly considered in a correcting function* $\alpha_\theta(\cdot)$ *introduced in Eq. 4 and 7 (instead of the constant smoothing parameter* α*); it is depending on the direction* θ^l *of the visual motion and defined as:*

$$\alpha_\theta(\theta^l) = \alpha \sqrt{\frac{1}{1 - e^2 \cos(\theta^l)^2}} \tag{9}$$

where $\alpha \in]0, 1]$ *and* $e = \sqrt{1 - \left(\frac{a}{b}\right)^2}$ *is the eccentricity of the ellipse, with* a *and* b *the width and the length of the photodiode, respectively. The second term of this equation increases the decay factor in the direction of the principal axis of the ellipse, rebalancing the representation of the moving edges in* $\mathbf{F}$*.*

2.3 APPLICATION TO CORNER DETECTION

In conventional frame-based vision, several techniques have been proposed that consist in determining points for which a measurement is locally optimal with respect to a criteria; in particular specific to corners. This measure can be computed by a cumulative process ([80]), using a self-similarity measure ([81]) derived from mathematical analysis (e.g. contour's local curvature ([82]), relying on an eigenvalue decomposition of a second-moment matrix ([83])) or selected as the output from a machine learning process ([84]).

In asynchronous event-based vision, [57] have proposed an algorithm based on the intersection of constraints principle (see [85]); which considers corners as locations where the aperture problem can be solved locally. Since cameras have a finite aperture size, motion estimation is possible only for directions orthogonal to edges. Fig. 9 shows the ambiguity due to the finite aperture. This can be written as follows: if $\mathbf{v}^n$ is the normal component of the velocity vector to an edge at time t at a location $\mathbf{p}$, then the real velocity vector is an element of the $\mathbb{R}^2$ subspace spanned by the unit vector $\mathbf{v}_t$, tangent to the edge at $\mathbf{p}$. This subspace is defined as $\mathcal{V}^1 = \{\mathbf{v} = \mathbf{v}_n + \alpha \mathbf{v}_t\}$ with $\alpha \in \mathbb{R}$. For a regular edge point, α can usually not be estimated. When two moving crossed gratings are superimposed to produce a coherent moving pattern, the velocity can be unambiguously estimated.

The geometry-based approach proposed in [57] consists in collecting planes, fitted directly on the event stream (as in [66] and Section 2.2.1) and considered as local observations of normal visual motions, around each visual

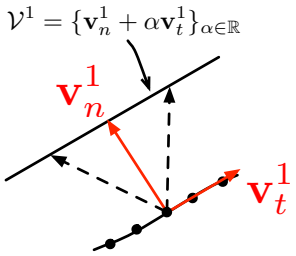


Figure 9: The aperture problem allows estimating only the normal component $\mathbf{v}_1^n$ of the velocity of events generated by an edge. The tangential component $\mathbf{v}_t^1$ is not recoverable. Any motion with the same component $\mathbf{v}_n^1$ induces the same stimulus. These motions define the real plane subspace $\mathcal{V}^1$. (extracted from [57])

event. This event is considered as a corner event (i.e. event generates at the spatiotemporal location of a corner) if most of the collected planes intersect as a straight line in $(XYT)^T$ reference frame, at a location temporally close to the event (see Fig. 10).

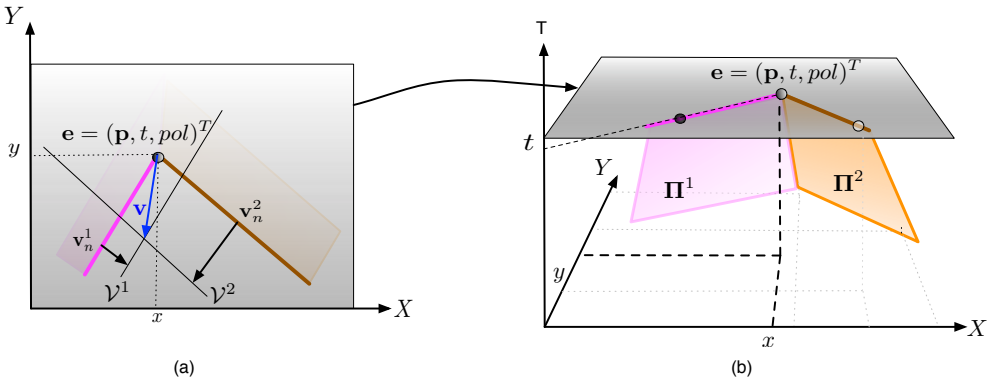


Figure 10: (a) An event e occurs at spatial location $\mathbf{p}$ at time t where two edges intersect. This configuration provides sufficient constraints to estimate the velocity $\mathbf{v}$ at $\mathbf{p}$ from the normal velocity vector $\mathbf{v}_n^1$ and $\mathbf{v}_n^2$ provided by the two edges. The velocity subspaces $\mathcal{V}^1$ and $\mathcal{V}^2$ are derived from the normal vectors. (b) Vectors $\mathbf{v}_n^1$ and $\mathbf{v}_n^2$ are computed by locally fitting two planes π^1 and π^2 on the events forming each edge over a space-time neighborhood. $\mathbf{v}_n^1$ and $\mathbf{v}_n^2$ are extracted from the slope of (respectively) π^1 and π^2 at $(\mathbf{p}, t)$. (extracted from [57])

2.3.1 Feature-based approaches

In the local approach, normalized $\mathbf{F}_{\mathbf{p},t}$ is the distribution of the normal velocities along the contours around the spatiotemporal location $(\mathbf{p}, t)^T$. In an ideal case illustrated in Figure 11, if this location corresponds to a corner location, $\mathbf{F}_{\mathbf{p},t}$ is null execpt around two velocity coordinates, $(v^n, \theta^n)^T$ and

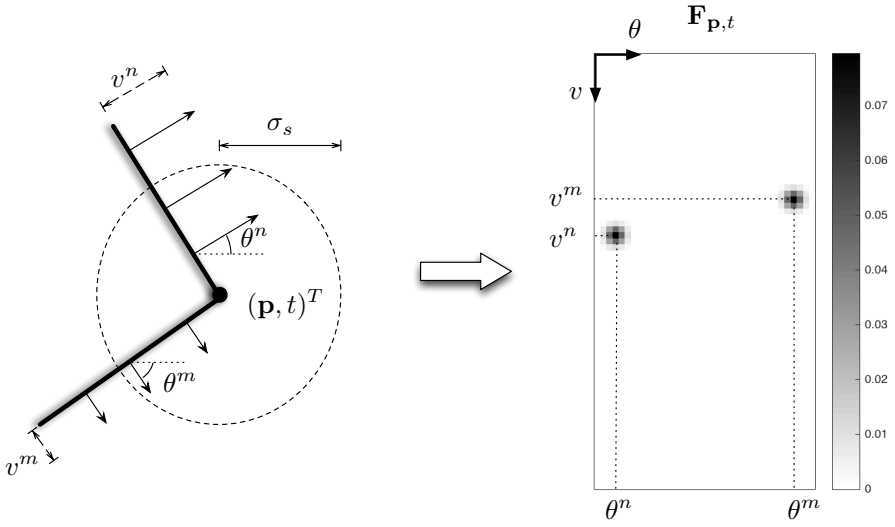


Figure 11: Illustration of the feature $\mathbf{F}_{\mathbf{p},t}$ (right figure) computed at the spatiotemporal location $(\mathbf{p}, t)^T$ of a corner (left figure) in an ideal case.

$(v^m, \theta^m)^T$, corresponding to both normal visual motions of the intersecting edges.

2-maxima based decision

As we can see in this Figure, detecting corners (or junctions) will consist in determining if at least two local maxima in $\mathbf{F}_{\mathbf{p},t}$ are present. We first propose an algorithm in order to find the two first maxima i n $\mathbf{F}_{\mathbf{p},t}$ consisting in:

1. finding the maximum F_{max} and its velocity coordinates $(v_{max}, \theta_{max})^T$ in $\mathbf{F}_{\mathbf{p},t}$,
2. inhibiting (set to zeros) all values in $\mathbf{F}$ for which the coordinates verify $|\theta^l - \theta_{max}| < th_\theta$, with $th_\theta = 20°$, and
3. finding the maximum (second maximum) $F_{2^{nd}max}$ and its coordinates $(v_{2^{nd}max}, \theta_{2^{nd}max})^T$ in $\mathbf{F}_{\mathbf{p},t}$ previously modified in step 2.

Finally, as an isoprobabilistic representation of the intersecting edges is assumed, both values of maxima, F_{max} and $F_{2^{nd}max}$, should be close at the location of a corner (the difference would be essentially due to noise). Then we propose as selection criterion (noted C_{2max}) to decide if a corner is present at $(\mathbf{p}, t)^T$:

$$C_{2max} = \frac{F_{2^{nd}max}}{F_{max}} > th_{C_{2max}} \tag{10}$$

with the threshold $th_{C_{2max}} \in]0, 1]$.

Velocity-constraint based decision

A second approach consists in considering each $(v^l, \theta^l)^T$ (or noted $(v_x^l, v_y^l)^T$ in a cartesian reference frame) as a velocity constraint $\mathcal{V}^l$ weighted by the

value $\mathbf{F}_{\mathbf{p},t}(v^l, \theta^l)$; verifying $(\mathbf{v}^l)^T\mathbf{v} = \|\mathbf{v}^l\|^2$, where $\mathbf{v} = (v_x, v_y)^T$ is the velocity of the corner.

A corner is present at location $(\mathbf{p}, t)^T$ if $\mathbf{F}_{\mathbf{p},t}$ gives rise to a real solution to the equation:

$$WA\mathbf{v} = W\mathbf{B} \tag{11}$$

where:

- $A = \begin{pmatrix} v_x^1 & v_y^1 \\ \vdots & \vdots \\ v_x^l & v_y^l \\ \vdots & \vdots \\ v_x^{N_v} & v_y^{N_v} \end{pmatrix}$, with $N_{\mathbf{v}} = N_v N_\theta$ the size of $\mathbf{F}_{\mathbf{p},t}$, i.e. the number of constraints,

- $\mathbf{B} = \begin{pmatrix} \|\mathbf{v}^1\|^2 \\ \vdots \\ \|\mathbf{v}^l\|^2 \\ \vdots \\ \|\mathbf{v}^{N_v}\|^2 \end{pmatrix}$ and $W = \text{diag}\left(\mathbf{F}_{\mathbf{p},t}(v^1, \theta^1), \ldots, \mathbf{F}_{\mathbf{p},t}(v^l, \theta^l), \ldots, \mathbf{F}_{\mathbf{p},t}(v^{N_v}, \theta^{N_v})\right)$.

Then the over-determined system can be solved if $M = (WA)^T WA$ has a full rank, meaning that its two eigenvalues have to be significantly large. This significance is determined with the selection criterion established in [86]:

$$\mathcal{C}_{const} = \frac{\det(M)}{\text{trace}(M)} > th_{\mathcal{C}_{const}} \tag{12}$$

with the threshold $th_{\mathcal{C}_{const}} > 0$.

Equation 11 is also solved with a least square minimization technique and solutions are considered as valid if $\mathcal{C}_{const}$ is greater than the threshold $th_{\mathcal{C}_{const}}$ usually set experimentally. Finally, a stream $\mathcal{S}^c$ of corner events (including features), noted $\mathbf{c} = (\mathbf{p}, \mathbf{v}, t, \mathbf{F})^T$, is outputted.

Remark 4 *In order to be robust to noise, weak values in $\mathbf{F}_{\mathbf{p},t}$ are inhibited (associated equations are filtered out of the system): if $\mathbf{F}_{\mathbf{p},t}(v^l, \theta^l) < th_{\mathbf{F}} F_{max}$ (with $th_{\mathbf{F}} \in]0, 1]$), then $F_{\mathbf{p},t}(v^l, \theta^l) = 0$.*

Remark 5 *With the 2-maxima based decision approach, a corner event stream can also be obtained; the velocities of the detected corners can be estimated in a similar manner using only both maxima's coordinates, without weighting them. Furthermore, while the second approach is based on a (unnatural) mathematical analysis, the first decision method is closer to a time-based neural implementation; it could be implemented as a coincidence detector between two (or more) events, denoting the*

two-first (or more) maxima, outputted by the leaky integrate-and-fire neural layer assimilated to the feature **F** *(see discussion at the end of Section 2.2.3).*

Note that neural networks have also been proposed in the literature ([87]) in order to solve similar systems of linear equations that are required in the velocity-constraint decision based method; VLSI implementations have even been proposed.

Remark 6 *Note that the computation principle is quite similar to the one proposed in [57]; most mechanisms involved (kernels, filters, selection criteria) have been designed and set in a similar manner, in order to allow comparison in the fairest way possible (see next Section). The methods differ from each other essentially by the selection process of the velocity constraints. Through a time-based weighting process, [57] considers only constraints along edges intersecting the evaluated event. The methods proposed in this article consider all the edges in a spatial neighborhood even if they are not perfectly intersecting themselves at the evaluated location; however the spatial Gaussian-based weights $w_s(\cdot)$ implicitly perform a heuristic selection of the spatially closer edges, i.e. the most probable intersecting edges. So even if the location of their detected corner events should be consequently less precise, they should be close to a real corner; this is verified in the results presented in the next Section.*

2.3.2 Evaluations

In order to evaluate the detectors, we reproduced one of the experiments proposed in [57], the one with the most quantitative evaluations. It consists into a swinging wired 3D cube shown to a neuromorphic camera (DVS, see Fig. 12).

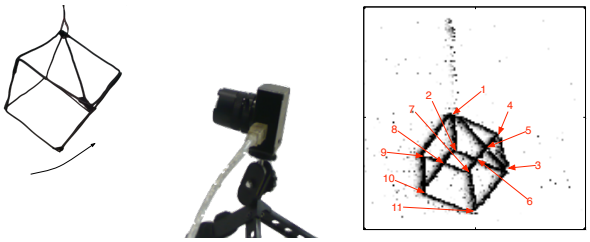


Figure 12: Illustration of the experiment: a swinging 3D cube is shown to a neuromorphic camera.

A complete accuracy evaluation, comparing the results obtained with the geometric-based method given in [57] and the methods proposed in this article, is provided in Fig. 13. The corner events parameters (spatial location and velocity) and the 11 corners' ones (obtained with the ground-truth) are compared using different measures of errors. Each corner event is associated to the spatially closest ground-truth corner's trajectory.

In order to propose a fair evaluation, the thresholds used in the different methods have been set in order to detect the same number of corner events (1500) and other algorithms' parameters have been set as the ones proposed in [57] (see Remark 6). The distribution of the corner events per corner's trajectory is shown in Fig. 15.a. We can observe that the distributions using the

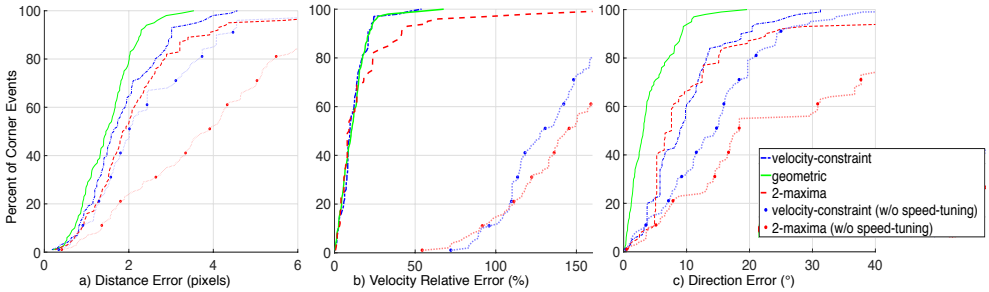


Figure 13: Precision Evaluation of the Corner Detectors; the green plain curves correspond to the results obtained with the algorithm proposed in [57]; the blue dash-dotted blue curves to the velocity-constraint based decision proposed in Section 2.3.1 and the red dashed curves to the 2-maxima based decision proposed in the Section 2.3.1. The blue and red dotted curves correspond to the respective feature-based approaches but without speed-tuned temporal kernels. The left figure (a) represents the spatial location errors of the corner events compared to the manually-obtained ground-truth trajectories of the corners; the middle one (b) the relative error about the intensity of the estimated speed and the right one (c) its error in direction. Accuracies (X-axis for the Figures) are given related to the considered percent (Y-axis) of the population of corner events detected with the different methods; e.g. with the method in [57], 80% of the corner events have a distance error in corner location less than 2*pixels* compared to the ground truth, see plain green curve in Fig. a).

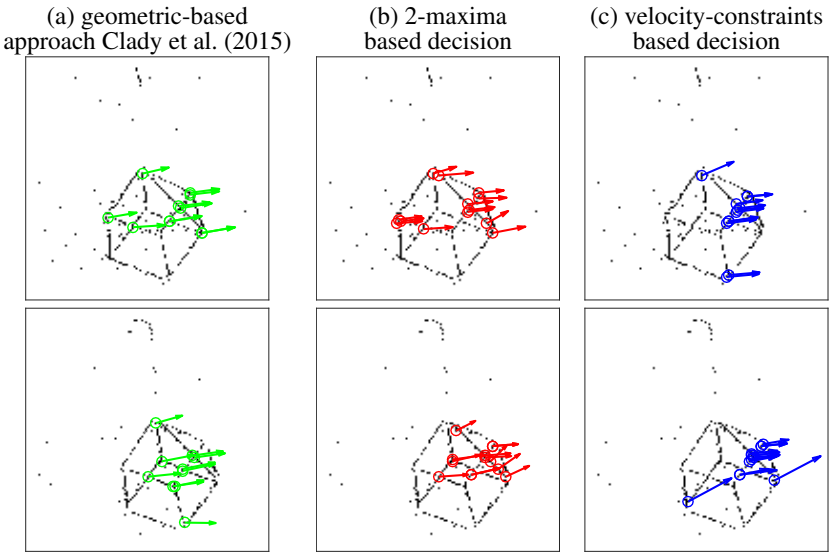
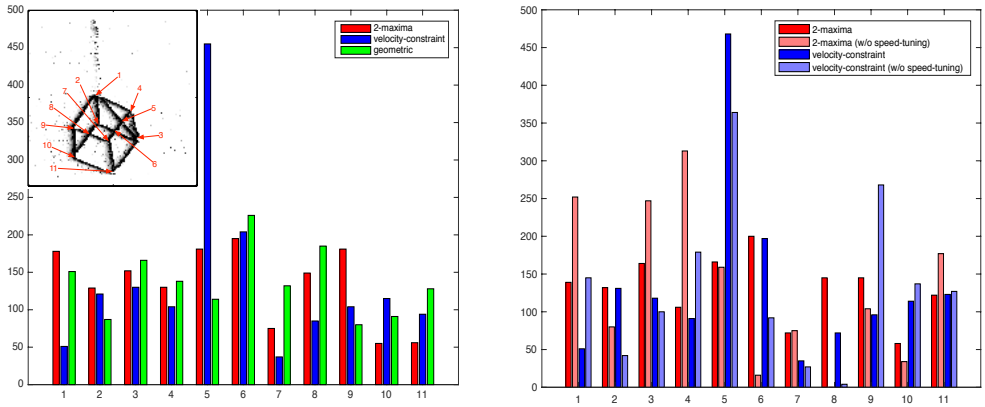


Figure 14: Snapshots of the results obtained for the 3 compared detectors, projecting in a frame the visual events (black dots) and corner events (circles, associated to vectors representing the estimated speeds) over two short time periods (1ms).

geometric-based and the 2-maxima decision based methods are closely similar. However the one obtained with the velocity-constraint decision based method is unbalanced, with a great number (close to the third of the corner events) of detections around a particular corner, corner number 5. This can

be explained by the fact that the proposed method is less spatially precise than the geometric-based one (cf. the curves in Fig. 13.a and Remark 6) and, as we can see in Fig. 14, the edges around this corner generated more events than the others because they are generated by "clean" intersecting edges, see Fig. 12, and then verifying well the ideal conditions for the optical flow estimation, and because it is a X-junction. It is not the case for the corners number 1, 7 and 11, for example; the high speed of the cube (close to 500pix.s^{-1}, i.e. inversely close to the precision of the event timings) and their badly shaped structures (they correspond to connections between the different wires constituting the cube) make their detection very hard due to the local bad quality of the event streams (in particular, there are numerous missing events as we can see in Fig. 15).

Remark 7 *Note that accuracy results in Fig. 13 concern median evaluations over the 11 ground-truth corners. Each corner is associated to the spatially closest ground-truth corner's trajectory. Each set of corner events (associated to a ground-truth corner) is sorted according to one of the evaluation criteria (type of errors). The Y%-most accurate corner events are then selected. Finally, the accuracy median value for this evaluation criterion is computed over all ground-truth's corners. So these evaluations are a priori not (or weakly) biased by these differences in distributions.*



(a) Comparison between the three evaluated detectors

(b) Comparison with or without (w/o) speed-tuned temporal kernels

Figure 15: Distributions of the detected event corners related to the labelled corners.

We can observe that the detectors proposed in this article are influenced by the quantification of the grid; especially in the Fig. 13.c representing the angular precision of the estimated speed direction. Indeed a lot of corner events have a direction-related precision close to 5°, the half of the direction-related interval length. The velocity-constrainst based decision method is less clearly influenced because it takes into account more elements in the feature (not only the elements with the maximal values, but also their neighboring elements) to estimate the speed.

In addition, Fig. 15.b shows the detections distribution for both feature-based methods, with or without speed-tuned temporal kernels. In the approaches without speed-tuning, the temporal decreasing factor τ has been

fixed as $\tau = \frac{1}{v_{mean}}$, where v_{mean} is the mean velocity computed over all corners and the stream duration (150ms). Without speed-tuning, some corners are not or not often detected, in particular corners number 6 and 8. They correspond to X-junctions between two intersecting edges with quite different dynamics, because generated by front and back wires. Furthermore, the accuracy performances for the approaches without speed-tuned temporal kernels are significantly lower than the ones with speed-tuned kernels, as shown in Fig. 13.

Finally, if we consider that a corner event detection is valid if the distance error is less than 3pixels, the geometric-based method generates only 2% of false alarms (with a median velocity error around 10% and a median direction error around 3° for the positive detections), while this rate rises to 8% and to 18% for the velocity-constraint decision and 2-maxima decision based methods, respectively (with a median velocity error around 10% and a median direction error around 8°, for both).

We have demonstrated that the proposed feature can be used (in its local approach) to detect corners in event streams. Even if the detectors are slightly less precise and more sensitive to the quality of the event streams than the other method proposed in the literature, our feature-based approaches are more efficient in terms of memory and computation loads.

Indeed the method in [57] requires to memorize the stream of the visual motion events (see Eq. 2) and spatiotemporal extrapolations of them (called "normal events") and operates quite complex computations between them. In the approach presented in this article, the visual motion events are integrated directly in the neighboring features, and corner detection related computations are operated only using the feature at the spatiotemporal location of the current event. We have measured important differences in terms of computation time between their different implementations; e.g. for the event stream used for the above evaluations, the feature-based approaches are approximatively 10 times faster. Table 1 presents the distribution of mean computation times obtained with the different approaches and over 10 repetitions (for 1500 detections). But as the method in [57] has been only implemented on Matlab (Matlab2015b), they should be taken with caution; it is indeed known that memory can be poorly managed on Matlab. Measuring the computation time without code lines dedicated to memory management (which is a crucial part of the method in [57]), the gain is still around 40%. While the geometric-based method is only envisioned in [57] for a real time implementation on massively parallel computers such as the SpiNNaker board (see [88, 61]), the feature-based approaches run in real-time on a standard computer (in C++ on a Intel Core i7-4790K @ 4GHz, using only one core and without any optimization such as integer arithmetic instead of floating point based computations, e.g. [89, 90]) for weakly complex visual scenes such as the one presented in this study.

Beyond this operational asset, the greatest strength of the proposed feature-based approaches lies in fact that they lead to a solution of the corner detection issue on event streams based on classical event-based neural network models (leaky integrate-and-fire neural network, coincidence detectors, etc.)

Methods	Total CT	% of CT OF estimation	% of CT feature computation	% of CT corner detection
velocity-constraint	76s.	16%	83%	1%
2-maxima	75s.	16%	83%	1%
geometric	828s.	1%	-	99%
geometric (w/o memory management)	132s.	9%	-	91%

Table 1: Distribution of mean computation times (CT) with the different approaches (estimated on Matlab2015b).

as it is highlighted in Section 2.2.3 and Remark 5.

2.4 APPLICATION TO GESTURE RECOGNITION

Human movement analysis is an area of study that has been quickly expanding since the 1990's (see [91, 92, 93]). The evolution and miniaturization of both computers and motion capturing sensors have made motion analysis possible in a growing set of environments. They have enabled numerous applications in robotics, control, surveillance, medical purposes ([94]) or even in video-games with the Microsoft's Kinect ([95]). However, the available technologies and methods still present numerous limitations, discouraging their use in embedded systems. Conventional time-sampled acquisition is very problematic when implemented in mobile devices because the embedded cameras usually operate at a frame-rate of 30 to 60 Hz: normal speed gesture movements can not be properly captured. Increasing the frame rate would result in the overload of the recognition algorithm, only displacing the bottleneck from acquisition to post-processing. Furthermore, conventional cameras and infrared-based methods are perturbed by dynamic lighting and infra-red radiations emitted by the sun (cf. [96]). Because they both require light-controlled environments, those technologies are unsuitable for outdoor use.

Asynchronous event-based sensing technology is expected to overcome several limitations encountered by state-of-the-art gesture recognition systems, in particular for battery-powered, mobile devices. These vision sensors, due to their near continuous-time operation, allow capturing the complete and true dynamics of human motion during the whole gesture duration. Due to the pixel-individual style of acquisition and pre-processing of the visual information, and in contrast to practically all existing technologies, they will be also able to support device operation under uncontrolled lighting conditions, particularly in outdoor scenarios (cf. [97]). Native redundancy suppression performed in event-based sensing and processing will ensure that computation can be performed in real time, while at the same time saving energy, decreasing system complexity.

Gesture recognition using neuromorphic camera has already been investigated by [63]. A stereo pair of DVS allows them to compute disparity in order to cluster the hand. Then, they use a tracking algorithm to extract

the 2D trajectory of the movement. Finally the trajectory is sampled into directions, and the obtained sequence of directions is fed to a HMM classifier. This approach uses event-based information only during the first step (extraction of the location of the hand). In addition, with this type of multi-steps architecture, a failure in a step could result in the failure of the whole system.

Here we propose to demonstrate that our feature can be used to detect and recognize more directly gestures. Hoof-like features (see Section 2.4.1) are derived from the feature matrix and provided to a classification architecture that performs simultaneously detection and recognition. It is based on hybrid generative/discriminative classifiers ([98]) in order to associate at each feature its probabilities to belong to the considered (hand) gestures or not, and these probabilities are integrated over time through a network of Bayes filters ([99]).

2.4.1 A more compact and invariant representation

In order to reduce the dimensionality of the feature (it is often required in machine learning, in order to address the "curse of dimensionality" issue) and to provide (global speed- and) scale-invariance property to the gesture representation, $\mathbf{F}$ can be transformed into a more compact representation, noted $\mathbf{h}$ ($\mathbf{h}_{\mathbf{p},t}$ or $\mathbf{h}_t$, in local or global approaches, respectively) and named hoof-like in reference to the Histogram of Oriented Optical Flow (HOOF) introduced by [100] in frame-based vision. This transformation consists in summing the intensities of the optical flow vectors with respect to their directions.

From the feature $\mathbf{F}$, $\mathbf{h}_{\mathbf{p},t} = \left[h_{\mathbf{p},t;1}, ..., h_{\mathbf{p},t;N_\theta}\right]^T$ can be easily obtained:

$$h_{\mathbf{p},t;i} = \sum_k v^k \mathbf{F}_{\mathbf{p},t}(v^k, \theta^i) \tag{13}$$

In the global approach, normalization (to sum to 1) makes the hoof-like feature globally speed- and scale-invariant. Figure 8.d represents the histogram of oriented optical flows computed globally on an event stream capturing a walking human (Fig. 8.a).

2.4.2 Classification Architecture

We propose a classification architecture where the problem is framed as a Bayes filter, that is estimating the probabilities of gestures recursively over time using incoming measurements, given as the hoof-like features $\mathbf{h}_{t_0:t_k} \in \mathcal{H}$ computed globally from every visual events $[\mathbf{e}_0, \mathbf{e}_k]$.

Then we note the state $g^i \in \mathcal{G}$, the gesture (numerated i, $i \in [1, K]$) that the user is currently performing. A state g^0 is added in $\mathcal{G}$, in order to consider the not-considered gestures or the instants while the user is not performing a hand gesture.

The camera observes the user's action and at each occurring feature estimates a distribution over the current state $g^i_{t_k}$:

$$p(g^i_{t_k} \mid \mathbf{h}_{t_0:t_k}) \tag{14}$$

where $\mathbf{h}_{t_k} \in \mathcal{H}$ is the observation of the gesture occurring at time t_k.

To estimate this probability, a time update and a measurement update are performed alternately. The time update updates the belief that the user is performing a specific gesture given previous information:

$$p(g^i_{t_k} \mid \mathbf{h}_{t_0:t_{k-1}}) = \sum_{g^j_{k-1} \in \mathcal{G}} p(g^i_{t_k} \mid g^j_{t_{k-1}}) p(g^j_{t_{k-1}} \mid \mathbf{h}_{t_0:t_{k-1}}) \tag{15}$$

The time update includes a transition probability from the previous state to the current state. As no-contextual information is available here, we assume that an user is likely to perform the same gesture, and at each timestamp has a large probability of transitioning to the same state:

$$p(g^i_{t_k} \mid g^j_{t_{k-1}}) = \begin{cases} \frac{1}{|\mathcal{G}|} + \frac{|\mathcal{G}|-1}{|\mathcal{G}|} \exp\left(-\frac{t_k - t_{k-1}}{\tau_g}\right) & \text{if } i = j \\ \frac{1}{|\mathcal{G}|} - \frac{1}{|\mathcal{G}|} \exp\left(-\frac{t_k - t_{k-1}}{\tau_g}\right) & \text{otherwise} \end{cases} \tag{16}$$

with τ_g set to 150ms, less than the half duration of shorter gestures. This assumption means that the gesture's certainty slowly decays over time, in the absence of corroborating information, converging to a uniform distribution (even if no event is observed).

The measurements update combines the previous belief with the newest observation to update each belief state, such as:

$$\begin{aligned} p(g^i_{t_k} \mid \mathbf{h}_{t_0:t_k}) &= \frac{p(\mathbf{h}_{t_k} \mid g^i_{t_k}) p(g^i_{t_k} \mid \mathbf{h}_{t_0:t_{k-1}})}{p(\mathbf{h}_{t_k} \mid \mathbf{h}_{t_0:t_{k-1}})} \\ &\propto p(\mathbf{h}_{t_k} \mid g^i_{t_k}) p(g^i_{t_k} \mid \mathbf{h}_{t_0:t_{k-1}}) \end{aligned} \tag{17}$$

In order to estimate $p(\mathbf{h}_{t_k} \mid g^i_{t_k})$, we propose a machine learning based approach to compute and select generative models for gesture. It is decomposed into two steps:

- For the first step, we collect hoof-like features computed while the users (included in the training database, see Section 2.4.3) performed a gesture g^i, $i \in [1, K]$. Then a k-means algorithm is applied on them in order to compute N candidate models, noted $\mathbf{m}^{g^i}$.
- The second step consists in selecting from these candidate models, the ones that are the most discriminative against hoof-like features collected from the rest of the training event streams; these last features have been computed during other considered gestures (g^j with $i \neq j$) or during other period times when users were not performing gestures. This selection is processed through a discrete Adaboost classifier.

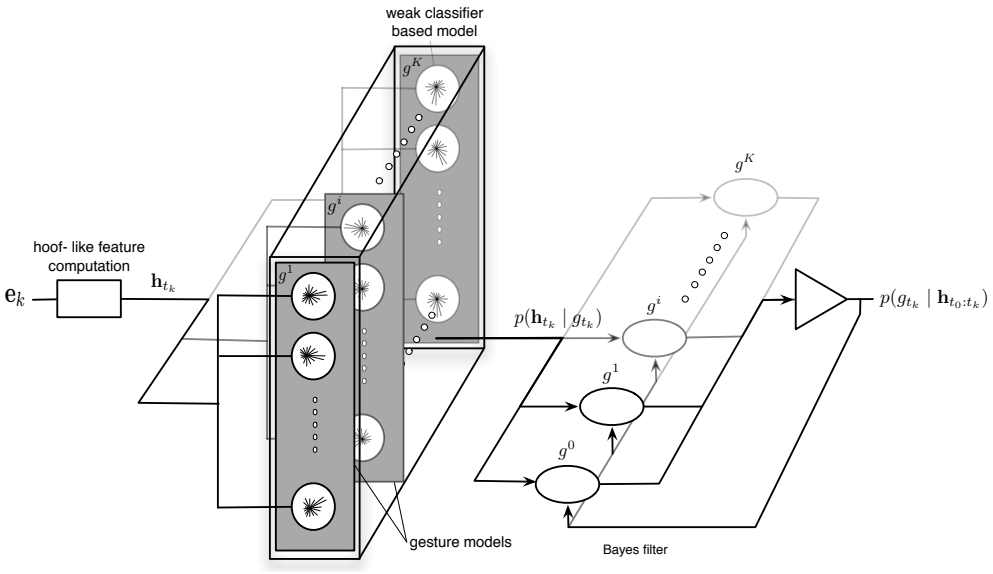


Figure 16: Gesture Recognition Architecture: for each occurring hoof-like feature $\mathbf{h}_{t_k}$, the distribution of probabilities noted $p(\mathbf{h}_{t_k} \mid g_{t_k})$, is estimated comparing the features to models computed and selected through an Adaboost-based learning process. Then the probabilities of gestures, noted $p(g_{t_k} \mid \mathbf{h}_{t_0:t_{k-1}})$, are estimated recursively over time.

Adaboost ([101]) is an iterative algorithm that finds, from a feature set, some weak but discriminative classification functions and combines them in a strong classification function:

$$B = \begin{cases} 1, \sum_{s=1}^{S} \lambda_s b_s \geqslant \frac{1}{2} \sum_{s=1}^{S} \lambda_s, \\ -1, \text{otherwise}, \end{cases} \tag{18}$$

where B and b are the strong and weak classification functions, respectively, and λ is a weight coefficient for each b. T is the threshold of the strong classifier B. The principle of the Adaboost algorithm is to select, at each iteration, a new weak classifier in favor of the instances (or features) misclassified by previous classifiers, through a weighting process attributing more influence to misclassified instances.

Note that a threshold value, noted th_B, can be defined (such as the condition in Eq. 18 can be written: $\frac{2}{\sum_{s=1}^{S} \lambda_s} \sum_{s=1}^{S} \lambda_s b_s \geqslant th_B$) in order to optimize a particular classification performance. During the learning step, its default value is 1; this means a classification frontier at the middle of the margin (see [102]). Increasing or reducing its value correspond to moving the frontier closer or further to the positive class, respectively.

In literature, discriminative training of generative models, as we propose here, has been shown as efficient learning methods in numerous applications as object or human detection ([103, 104, 105]), face or character recognition ([106, 107]) or for medical purposes ([108, 109]). The proposed classifier based on the training and the selection of generative models in a discriminative way, combines indeed the main characteristics of discriminative and

generative approaches: discriminative power and generalization ability, respectively. The latter is in particular very important in our application, when a weak amount of labelled training data is available, see Section 2.4.3.

Following the framework described in [110], we propose to design weak classifiers as generative ones, associated to each candidate models $\mathbf{m}_s^{g^i}$ ($s \in [1, N]$):

$$b_s^i = \begin{cases} 1, \text{ if } f(\mathbf{h}, \mathbf{m}_s^{g^i}) = \exp\left(-\frac{d(\mathbf{h}, \mathbf{m}_s^{g^i})^2}{\theta_s^{g^i}}\right) \geqslant \frac{1}{2} \\ -1, \text{ otherwise,} \end{cases} \tag{19}$$

where $d(\cdot, \cdot)$ is the Euclidean distance and $\theta_s^{g^i}$ parametrizes the likelihood function f and is computed at each iteration of the algorithm through a maximum-likelihood estimation (taking into account the weights attributed to features).

During training, Adaboost based algorithm tends to select iteratively the most discriminative and complementary models for each gesture. We limit the number of selected models, such as the relative difference between F-measure (computed on training database, see Section 2.4.3) obtained at the corresponding iteration is superior or equal to 95% of its maximum (obtained with a greater number of iterations of Adaboost algorithm). Let us remind that F-measure is defined as $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. Optimizing it means also to determine a number of models for which an acceptable compromise between precision (the ratio of positive detections to instances belonging to performed gestures) and recall (the ratio of positive detections to all instances detected as belonging to gestures) is reached.

The probability $p(\mathbf{h}_{t_k} \mid g_{t_k}^i)$ is then estimated as proportional to a measure ($\in [0, 1]$) operated between the hoof-like feature and the set of selected models (applying a sigmoidal function to the output of the strong classifier):

$$p(\mathbf{h}_{t_k} \mid g_{t_k}^i) \propto \mathcal{L}(\mathbf{h}_{t_k}, g^i) = \frac{1}{1 + \exp\left(\frac{2}{\sum_{s=1}^{S^i} \lambda_s^i} \sum_{s=1}^{S^i} \lambda_s^i b_s^i - th_B^i\right)} \tag{20}$$

with $i \in [1, K]$ and th_B^i is the threshold obtained optimizing the F-measure. The probability associated to not-considered gesture (or no-gesture), noted g^0, is then defined as:

$$p(\mathbf{h}_{t_k} \mid g_{t_k}^0) \propto 1 - \max_{i \in [1,K]} \left(\mathcal{L}(\mathbf{h}_{t_k}, g^i)\right) \tag{21}$$

Figure 16 presents the obtained classification architecture. Finally a gesture's class G_{t_k} at each time is attributed from the distribution of probabilities, defined as:

$$G_{t_k} = \text{argmax}_{i \in [0,K]} \left(p(g_{t_k}^i \mid \mathbf{h}_{t_0:t_k})\right) \tag{22}$$

Remark 8 *Even if our implementation is based on a learning process not directly related to neural approaches (essentially due to the limited size of the database), we can observe that the resulting classification architecture could be fully implemented in an event-based framework. Through a rate-coding model, hoof-like features could be computed and transmitted from the leaky integrate-and-fire neural network, corresponding to the feature computation, as evoked in Section 2.2.3, to neural networks performing their comparison with gesture models (considering maybe another distance than the Euclidean one used here) and outputting positive events when they match; these positive events corresponding to the weak classifier responses (b_s^i). The coefficients λ_s^i would be then assimilated to synaptic weights. The other operations, in particular involved in Bayes filters, would correspond to feedback lines and basic mathematical operations that can be modeled using precise timing and event-based paradigms as demonstrated in [111].*

2.4.3 Results

Experimental Protocol

The protocol assumes that the users performed gestures in front of the camera. Event streams (using the ATIS camera) have been collected with 9 users (young and middle-aged people working in the laboratory). All users are right-handed but the database could be extended to left-handed users by mirroring the sequences horizontally.

(a) Example of a hand gesture performed in front of the camera

(b) ATIS camera embedded on a smartphone

Figure 17: Illustration of the targeted human-machine interaction.

The hand is moving at a distance around 30cm from the camera, approximatively. Note that this distance has been determined to ensure that the hand is fully viewed by the camera (see Figure 17.a) considering the current optic lens (this distance should be reduced when a wider-angle lens will be implemented). Each gesture is repeated five times by each user, varying the hand speed.

Six gestures have been defined and correspond to a dictionary of coarse gestures; the gesture is defined by the global motion of the hand (hand moving to the left, to the right, upward, downward, opening or closing). These

gestures could match with the main controls we could intend to execute interacting with a smartphone or a tablet (navigating in a menu or a list, selecting/unselecting an object or an application), i.e. the targeted application (see Fig. 17.b). Furthermore they constitute a dictionary for more complex gestures, successively combining these movements. In Fig. 18, an iconic representation of these coarse gestures is presented in the second column.

The training database is composed of the event streams collected with five users and the test database with the four other ones. During the evaluations (see next Section), a cross-validation is performed ten times (presented evaluations are the obtained mean values), putting randomly the users in the training or test databases. $30,000$ hoof-like features, computed on the training streams, are collected randomly and equitably in the time periods when gestures are performed (including the not-considered gestures or no gesture class) to train the Adaboost classifiers with a *one-vs.-all* strategy. An equal quantity is again randomly selected for the F-measure based optimization process and the selection of the number of models. 600 candidate models per gesture have been computed using k-means algorithm. The characteristics of the hoof-like features are the same as described in Section 2.2.2 ($N_\theta = 36$, etc).

A gesture is considered as detected when the duration of a time period with classified gestures ($G_{t_k} \neq 0$ in Eq. 22) is over $300ms$. This detection is counted as positive if this time period overlaps the manually labelled ground truth (with an overlap ratio superior to 0.5) .

Evaluations

Figure 18 represents the considered gestures and the models selected by Adaboost during a learning process (see Section 2.4.2). We can observe that the number of selected models is relatively weak (3 or 4). This means that the hoof-like features are able to represent well the gestures despite their (speed- and user-related) variability, mostly thanks to its speed- and scale-invariance property.

Another observation concerns the "shape" of the feature models. For most of them, they match well to the iconic representation of the corresponding motion; for example, for the motions to the left and to the right, most speed vectors are oriented to these respective directions, etc. However, some singularities have to be explained considering not only the global motion but also the directions of the principal contours of the human parts (hand, finger and arm) involved in the hand movement. For the opening hand motion, models obtained at iterations 1 and 2 highlight the motion of the thumb, for which the moving contours are prevalent in the feature. For the downward motion, the contours of the arm are too prevalent (see models obtained at iterations 2 and 3) because the camera viewed the user's bust (see Fig. 17).

In terms of detection performance, we obtained a mean precision of 91% and a mean recall of 83% (F-measure = 0.85) which confirm the great discrimination power of the proposed feature. Note that the F-measures obtained during the optimization (to determine th_B and the number of models)

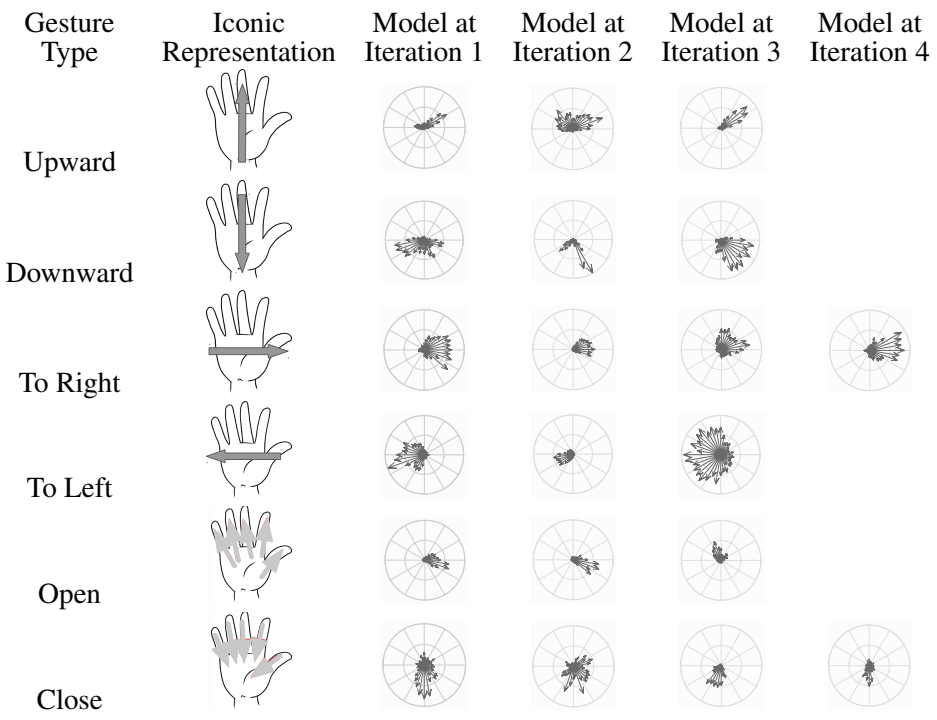


Figure 18: Iconic representations (second column) of the gestures (first column) and corresponding models selected by the Adaboost-based machine learning process.

are around 0.75. The greater value obtained at the final output highlights the filtering action of the Bayes filters.

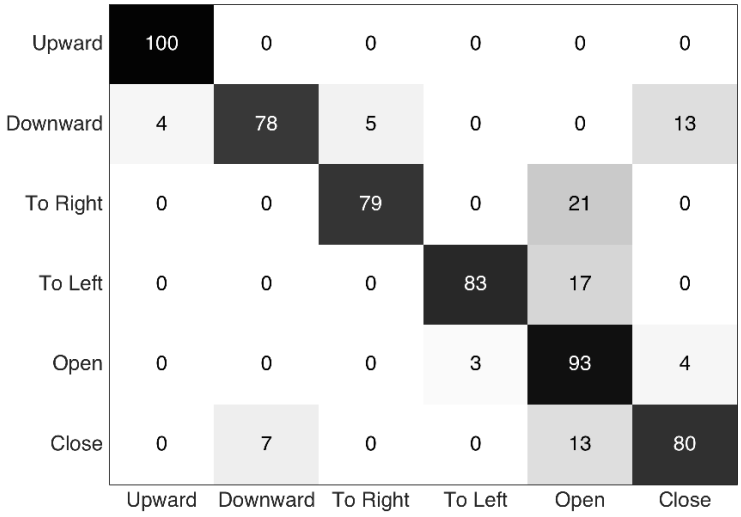


Figure 19: Confusion matrix (expressed in percent) showing the recognized gestures (columns) related to the performed gestures (lines), among the positive detections.

Finally the confusion matrix given in Fig. 19 shows us the recognized gestures among the positive detections. The downward and closing hand gestures are obviously a little confused because the similarity of the hand's

and the fingers' motions, respectively. The confusion of other gestures with the opening hand is probably due to the fact that the gesture is hard to detect, probably because the larger proportion of the movement involved the other fingers than the thumb and their moving contours generated few visual events (because in folded positions; the finger-skin *vs.* palm-skin contrast changes are weakly captured, see Remark 2). Indeed, in order to optimize the F-measure, the proposed process tends to select a low threshold compared to others (3 or 4 times lower); this means that classification frontier defined for this gesture tends to include other gestures. Hence, these gestures are sometimes misclassified as opening hand.

In further developments, we expect to improve these performances combining this global feature with locally computed ones, taking into account their relative spatio-temporal relationships. This should help us to better distinct the global motion of the hand and the local motions of the fingers, and hence better detect and categorize gestures.

2.5 CONCLUSION AND DISCUSSION

In this article, we have proposed a motion-based feature for event-based vision. It consists in encoding the local or global visual information provided by a neuromorphic camera, in a grid-sampled map of optical flow. Collecting optical flow (or visual motion events) computed around each visual event in a neighborhood or in the entire retina, this map represents their current probabilistic distribution in a speed- and direction-coordinates frame.

Two event-based pattern recognition frameworks have been developed in order to demonstrate its usefulness for such tasks. The first one is dedicated to detection of specific interest points, corners. Two feature-based approaches have been developed and evaluated. Formulated as an intersection of constraints issue, this fundamental task in computer vision can be resolved operating with the information encoded in the proposed local feature. The second one consists in a hand gesture recognition system for human-machine interaction, in particular with mobile devices. More compact and scale-invariant representations (called hoof-like features) of the motion observed in the visual scene, are extracted directly from the global version of the proposed feature, and feed a classification architecture, based on a discriminative learning schema of gestures' generative models and framed as a Bayes filter. Evaluations show that this feature has sufficient descriptive power to solve such pattern recognition problems. Other extensions or derivations of the proposed feature can be also envisioned in further developments, in order to address other pattern recognition issues. For example, summing the elements of the feature, with respect to their directions and without weighting them by corresponding speed, will result into another compact form, similar to the hog (histogram of oriented gradients) feature proposed by [112]. This feature and its derivations have been demonstrated as very efficient for many pattern recognition tasks in frame-based vision. To evaluate it in event-based vision would require to design event-based and dedicated classification architecture(s).

It is interesting to notice that our motion-based feature allows us to detect features defined by "static" properties, i.e. corners, and recognize dynamic actions, i.e. gestures, in visual scenes. All required information for both tasks are provided by a local computation of optical flow; this information is precisely encoded in the primary area (V1) of the visual cortex via the selectivity of V1 neurons. We underline also that the proposed frameworks are fully incremental and could be implemented as event-based neural networks, in particular thanks to speed and direction coordinates frame based representation of the visual motion information.

Such polar coordinate frame based representations have been already investigated for computer vision; e.g. based on bank of Gabor filters, using whether synchronous frame-based ([113, 114, 115], etc.) or asynchronous event-based ([60]) visual information. Works about natural image statistics ([116]) showed that similar decompositions of visual information emerge naturally from independent component analysis applied on patches collected on natural images. Recently, a work in [117] encoding more directly local event streams as local spatiotemporal surfaces ([7]), showed that an unsupervised learning process applied on a relatively large database acquired with a neuromorphic camera, leads to a similar result: basic and local feature extractors coding contours' speed and direction. Moreover, other works ([118, 100, 119], etc.) in frame-based vision have shown that optical flow is a valuable information to encode in features for pattern recognition tasks.

In addition, the work presented in this article supports the proposition that optical flow's speed and direction based grid is not only a powerful manner for encoding visual information in pattern recognition tasks, but it plays also a key role at a computational level when dealing with asynchronous event-based streams. Indeed we have shown that, to compute the distribution of optical flow along current edges, we need to take into account their respective dynamics, in order to ensure that the moving edges are equitably represented in the feature (whatever their own dynamics). The discretization of the visual motion information into the proposed speed- and direction-based grid allows us to incorporate directly the required speed-tuned temporal kernels in the structure of the computational architecture computing the feature. We have in addition proposed that this architecture can be implemented as a leaky integrate-and-fire neural layer, wherein neurons have then speed-tuned integration times; so it could be further integrated as the first layer in a spiking neural network using back-propagation based deep learning technique, as the one recently proposed by [62] wherein LIF neurons are also used.

Finally, in the asynchronous event-based multilayer architectures proposed recently in [117, 7], the integration times are tuned as increasing at higher layers. In addition, in our gesture recognition architecture, we have set the integration time in Bayes filters regarding the gesture durations, not the dynamics of the visual information. Further investigations could address the following issue: when (or at what level in hierarchical models) the integration times should be tuned not regarding the dynamics of the perceived information, but other temporal considerations or dynamics, maybe related

to a targeted task or action, or maybe related to other perceptive, learning or memory functions.

3 EVENT-BASED GESTURE RECOGNITION WITH DYNAMIC BACKGROUND SUPPRESSION USING SMARTPHONE COMPUTATIONAL CAPABILITIES

In this chapter, we introduce a framework for dynamic gesture recognition with background suppression operating on the output of a moving event-based camera. The system is developed to operate in real-time using only the computational capabilities of a mobile phone. It introduces a new development around the concept of time-surfaces. It also presents a novel event-based methodology to dynamically remove backgrounds that uses the high temporal resolution properties of event-based cameras. To our knowledge, this is the first Android event-based framework for vision-based recognition of *dynamic* gestures running on a smartphone without off-board processing. We assess the performance by considering several scenarios in both indoors and outdoors, for static and dynamic conditions, in uncontrolled lighting conditions. We also introduce a new event-based dataset for gesture recognition with static and dynamic backgrounds (made publicly available). The set of gestures has been selected following a clinical trial to allow human-machine interaction for the visually impaired and older adults. We finally report comparisons with prior work that addressed event-based gesture recognition reporting comparable results, without the use of advanced classification techniques nor power greedy hardware.

3.1 INTRODUCTION

We focuse on the problem of gesture recognition and dynamic background suppression using the output of a neuromorphic asynchronous event-based camera (Fig.20) connected to a mobile phone [120]. The system does not rely on off-board resources. Event-based cameras are scene driven, and their power consumption depends on the amount of recorded data (typically 1-10mW). They hold the promise of low computational costs while operating at high temporal scales. However, there has been no development of a proof of concept using these properties in the context of edge computation. In this chapter, we introduce a working prototype of a smartphone event-based application. We chose the popular task of vision-based gesture recognition and dynamic background suppression. These are good targets to make use of the dynamic properties of event-based sensors. We chose to use a scalable machine learning architecture relying on the concept of time-surfaces introduced in [7] and extended it to operate on the limited available computational resources. The system has been designed to operate on each incoming

event rather than creating frames from the output of the sensor to then send them to a GPU.
Compared to previous event-based approaches that tackled the problem of gesture recognition, we emphasize the importance of using the information carried out by the timing of past events to obtain a robust low-level feature representation to avoid binning events into frames. We also address the difficult problem of dynamic background suppression by introducing a novel low power event-based technique operating in the temporal domain. This technique goes beyond existing background suppression methodologies. It uses the properties of data-driven acquisition and its high temporal resolution to segment a scene by setting a relation between depth and relative activity, thus allowing the foreground and background to be differentiated.
We also introduce a new dataset of gestures (*NavGesture*) recorded using an event-based camera and available for public download. The neuromorphic field still lacks datasets that take full advantage of the precise timing of event-based cameras. Available datasets such as N-MNIST and N-Caltech101 [121] are recording scenes where dynamics are artificially introduced. Even true neuromorphic datasets such as Poker-DVS [122] or N-Cars [123] contain limited *intrinsic* dynamic properties that could be used for classification. We intend to observe objects that can be classified using only their dynamic properties (or motion) and not from their spatial distribution. As an example, if one considers the N-Cars [123] database, most objects appear as "flashes" that provide a snapshot of the object to be recognized. The Dvs-Gesture dataset [13] fulfills the requirement of having dynamic properties, however the camera is set static with the same centring for all samples with no activity in the background. The American Sign Language dataset, ASL-DVS [124] offers various centring and scales but aims to recognizing hand postures and also lacks dynamic properties. The proposed dataset (*NavGesture*) is a new step towards bridging the gap between laboratory-recorded datasets and everyday real situations. It features a set of six dynamic gestures, with heterogeneous centring and scaling, and was recorded a moving event-based camera, both in indoor and outdoor environments.

3.1.1 Gesture Recognition on Mobile Devices

Gesture recognition on mobile devices is a quickly expanding field of research that uses a variety of sensors and methods [125, 126, 127]. While resource-constrained devices such as smartphones disallow the use of certain technologies requiring high energy consumption such as vision-based depth (RGB-D) sensors, current mobile phones have a wide variety of built-in sensors. Several techniques use: phone speakers [128], inertial sensors [129, 130, 131] or proximity sensors [132, 133]. It is worth noticing that Won et al. [134] propose to use a neuromorphic camera as a proximity sensor instead of the conventional infra-red sensitive photo-diode. Other techniques use external components such as: e-gloves [135], radio-frequency chips [136] and even an IMU for teeth "clicks" recognition [137].
Smartphones also use standard RGB cameras, allowing vision-based recog-

Figure 20: A neuromorphic camera (an ATIS) (B) is plugged into a smartphone (A) using an USB link (C), allowing mid-air gesture navigation on the smartphone.

nition. As pointed in [138], dynamic gestures must be captured at high frame rates in order to avoid motion blur and in some cases even missing a gesture. However, processing high frame rates video data in real time on a smartphone is computationally challenging if not impossible. This might explain why most if not all of the vision-based gesture recognition methods running on smartphones without off-board processing are only applied to static gestures (hand poses) [139, 140]. The only vision-based dynamic gesture recognition method for smartphone we found is proposed by Rao et al. [141]. However, no proof of concept operating on a smartphone has been developed as the system has only been simulated on a resource-capped standard computer. Furthermore vision-based methods require to segment the hand from the background. This is often solved either by background pre-sampling [142] or by using skin color calibration [143, 144]. We will shortly show that this can be performed differently if one considers the high temporal resolution of event-based cameras.

3.1.2 Gesture Recognition using Event-based Cameras

Neuromorphic cameras coupled with event-based processing open new perspectives for resource management as both computation and memory can be allocated only to active parts of a visual scene. In the past few years

a large number of works tackled computer vision problems using event-based cameras while keeping in mind the necessity of avoiding at all costs the temptation to generate frames from the sensor's output, to cite a few: optical flow estimation [145], high-speed tracking [146, 147, 148], object classification [149, 150, 151], 3D reconstruction [152] or pose estimation [153].
Generating images from the output of event-based cameras to take advantage of decades of standard computer vision research is becoming a popular stream of research [154, 155, 156, 157]. This has lead to the development of pipelines that convert conventional frame-based datasets into events either using hardware [121, 158, 159] or software [160]. These data are then often converted back into frames in order to use frame-based techniques such as CNNs. There is currently a need to carry out research on event-by-event processing to take full advantage of all the properties of neuromorphic vision sensors [161, 162]. These sensors cannot only be used to generate high frame rates or high dynamic range images as one loses all advantages of the sparseness and low computation power associated to event-based acquisition.
To our knowledge, the first gesture recognition system using a Dynamic Vision Sensors (DVS) is the Rock-Scissor-Paper game from Ahn et al. [163], which detected the final static hand pose using event activity. Samsung has developed several gesture recognition systems. In early experiments, they proposed to use Leaky Integrate-and-Fire (LIF) neurons to correlate space-time events in order to extract the trajectory of gestures, using a stereo-pair of DVS in [164, 63]. This method is also adapted to track a finger tip using a single DVS [165], and event activity rate is also used to discriminate finger tip movements from hand swipes. Samsung also proposed to use the Adaptive Resonance Theory (ART) for continuous gesture recognition, first with HMM [166], then with CNN [167]. In parallel to the trajectory extraction approaches, global motion-based features were proposed. Kohn et al. [168] proposed a motion-based analysis of body movements using the relative event activity accumulated into 40 ms frames, while Lee et al. [169] used pseudo optical-flow. To cope with varying speeds, Clady et al. [170] proposed a motion-based feature that decays depending on the speed of the optical flow. Two end-to-end neuromorphic systems for gesture recognition have been proposed in recent years. The first one used the SpiNNaker neuromorphic board [171] and the second was implemented by IBM Research on the TrueNorth neuromorphic chip [13]. However, both systems bin events into frames at some point in order to use a CNN for classification. Along with their implementation IBM has also released the DvsGesture dataset, which has become widely used in the neuromorphic community. It has been used in multiple papers: spatio-temporal filters that feed a CNN [14], SNN [172, 173] and a PointNet adaptation [174].
Sign Language recognition has also been investigated but with a focus on static hand postures using events-to-frame techniques [175] or a graph-based CNN [124]. Chen et al. [162] proposed a new representation called *Fixed Length Gist Representation* (FLGR), mapping events to a higher dimensional feature. All presented methods used data from a static neuromorphic camera, with no background clutter. Furthermore, centring and scaling is in general the same except for [124]. The only work to our knowledge that addresses cluttered and dynamic backgrounds is the hand detection method

proposed by Samsung [176]. Unfortunately, they did not release their dataset that also include one to several subjects per clips. Also, it is worth mentioning that almost all presented works use at some point an events-to-frame conversion such as temporal or index binning, pixel spike rate or global memory surfaces. The only methods that process events in an event-based manner are scarce: [165, 169], Clady et al. [170], SLAYER [172] and FLGR [6].
In this work, we will consider more general scenarios offered by a moving camera that induces numerous new issues to solve such as: a higher number of emitted events, heterogeneous centring and scaling, unwanted shaking and important background clutter. Eliminating the background is an important step for event by event processing. Kyung et al. [177] proposed a background suppression method for neuromorphic cameras, but converted events to frames. Our approach is purely event-based and drastically contrasts from any existing background removal algorithm as it uses only the timing of events and it does not rely on conventional approaches such as: code-books [178], probabilistic approaches [179], sample-based methods [180], subspace-based techniques [181] or even deep learning [182].

3.2 METHODS

3.2.1 Dynamic Background Suppression

The Dynamic Background Suppression (DBS) uses the simple idea that the closer an object is to the camera, the more events it will generate as its apparent motion will be more important than a farther object. From this property it is possible to link the relative local activity within the focal plane to depth. A low event relative activity can be associated to the background and hence dismissed, whereas relative high activity regions could correspond to the foreground. Although the technique could be applied to each pixel, we will estimate the relative activity considering portions of the focal plane that will be divided into a grid of cells.

Let each cell c be composed of a set of pixels where activity is expressed by A_c. For each incoming event $e_k = (\boldsymbol{x_k}, t_k, p_k)$ emitted by a pixel belonging to a cell c, we can apply the following update of its activity A_c as:

$$A_c \leftarrow A_c \cdot \exp(-\frac{t_k - t_c}{\tau_b}) + 1 \tag{23}$$

where t_k is the time-stamp of the current event e_k, t_c the last time c has been updated, and τ_b is a decaying time-constant.

We can then compute the average activity $\overline{A}$ of a all cells. An incoming event $e_k = (\boldsymbol{x_k}, t_k, p_k)$ belonging to c is sent to the machine learning module only if:

$$A_c \geqslant \max(\alpha\overline{A}, A_T) \tag{24}$$

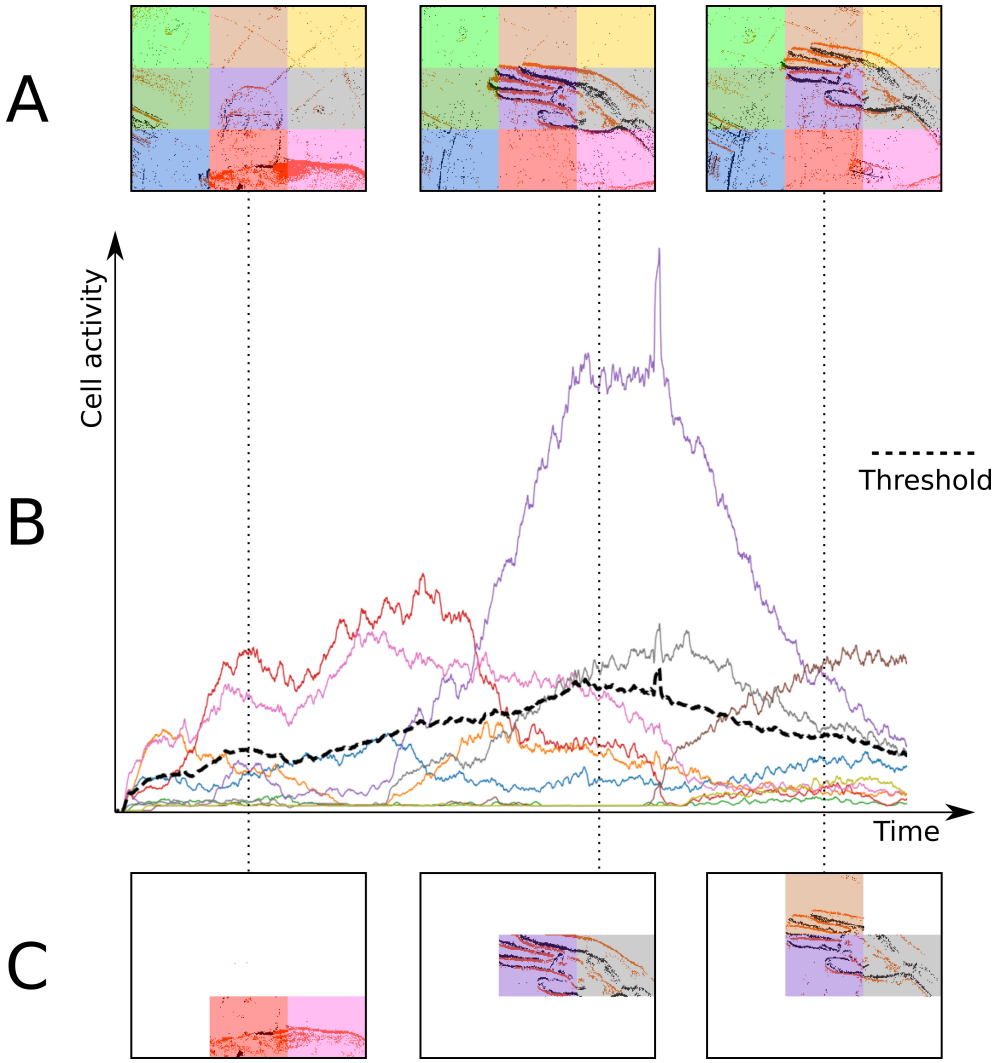


Figure 21: Operating principle of the Dynamic Background Suppression (DBS). (A) A gesture is performed in front of the camera, which pixel array is divided into cells. (B) Each cell has its own activity counter that decays over time. (C) Only cells with their activity greater than the mean activity (black dashes) of all cells can spike.

where α is a scalar to set the aggressiveness of the filter, and A_T is a threshold for minimum foreground activity. The activity of a cell and the threshold $\overline{A}$ are computed for each incoming event, which enables or disables a given cell at the temporal resolution of incoming events. Cells with a low activity are considered as background and are prevented from emitting events. In principle each time a cell is updated the general mean activity has to be updated. Events are timed at the µs and are orders of magnitude faster than any conventional urban real scene dynamics. The mean activity can then be updated at much lower temporal scales set experimentally according to the computation power available and perhaps the situation (one can infer acceleration from the built-in IMU). The proof of principle of the technique is shown in Fig. 22.

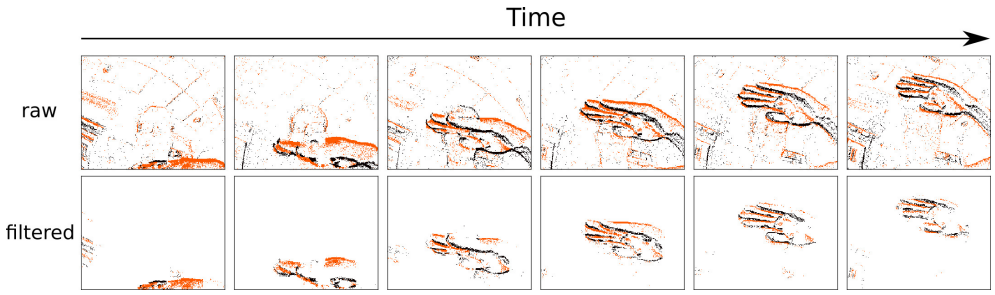


Figure 22: Denoising example of a gesture clip from the NavGesture-walk data-set. The presented gesture is a "swipe down". Top row is the raw stream of visual events, and the bottom row is the denoised stream, at the output of the 3rd stage of the cascade presented here. Each snapshot from the top row is made of 10,000 events, and bottom row contains only the kept events of those 10,000. "ON" events are orange, "OFF" events are black. The filtering lead to the removal of 83.8% of all events. Even after removing this many events each gesture is still easily recognizable by the human eye.

3.2.2 Time-surfaces as spatio-temporal descriptors

A time-surface [7] is a descriptor of the spatio-temporal neighborhood around an incoming event e_k. We define the time-context $T_k(\boldsymbol{u}, p)$ of the event e_k as a map of time differences between the time-stamp of the current event and the time-stamps of the most recent events in its spatial neighborhood. This $(2R+1) \times (2R+1)$ map is centered on e_k, of spatial coordinates $\boldsymbol{x_k}$. The time-context can be expressed as:

$$T_k(\boldsymbol{u}, p) = \{t_k - t \mid t = \max_{j \leqslant k} \{t_j \mid \boldsymbol{x_j} = (\boldsymbol{x_k} + \boldsymbol{u}),\ p_j = p\}\} \tag{25}$$

where $\boldsymbol{u} = [u_x, u_y]^T$ is such that $u_x \in [\![-R, R]\!]$ and $u_y \in [\![-R, R]\!]$.

Finally, we obtain the time-surface $S_k(\boldsymbol{u}, p)$ associated with the event e_k, by applying a linear decay kernel of time-constant τ to the time-context T_k:

$$S_k(\boldsymbol{u}, p) = \begin{cases} 1 - \frac{T_k(\boldsymbol{u}, p)}{\tau}, & \text{if } T_k(\boldsymbol{u}, p) < \tau \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

S_k is a low-level representation of the local spatio-temporal neighborhood of the event e_k.

Discarding time-surfaces. A time-surface can be computed for each new incoming event, but would generate overlapping time-surfaces and introduce redundancy. As the event-based camera performs native edge extraction, we must ensure that a sufficient number of events to form a full contour are taken into account. Therefore, time-surfaces must be kept only if they contain enough information, which can be ensured using the following heuristic:

$$card(\{(\boldsymbol{u}, p),\ T_k(\boldsymbol{u}, p) < \tau\}) \geqslant 2R \tag{27}$$

3.2.3 Event-based Hierarchical Pattern Matching

Following the principle of using deep multiple temporal and spatial scales introduced in HOTS [7], incoming visual events are fed to a network com-

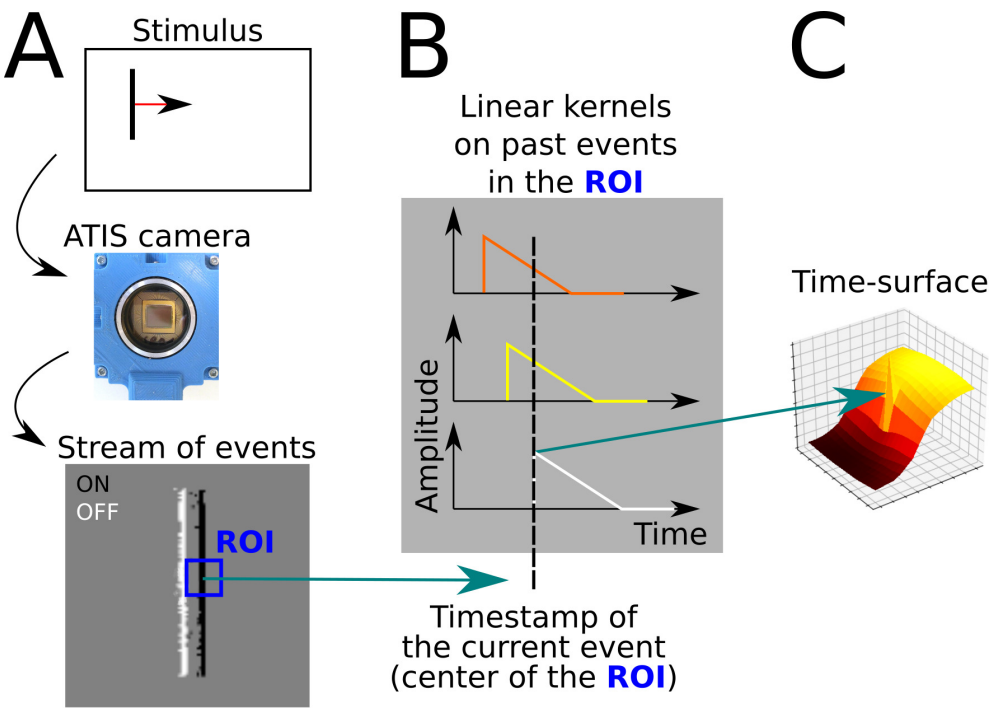


Figure 23: (A) A moving vertical bar is presented to the event-based camera, which output a stream of visual events. The edges of the bar are ON (white) and OFF (black) events. A ROI is defined around the current event (blue square). (B) The time-stamps of visual events contained the ROI are decayed using a linear kernel. (C) The resulting extracted time-surface, that encodes both the contour orientation and the dynamic of the motion.

posed of several layers. As events flow into the network, only their polarities are updated on successive "feature planes". Polarities in the network correspond to learned patterns or elementary features at that temporal and spatial scale. However, as time-surfaces can be discarded, the network output stream contains less events than the input stream, which is an important property that builds on the native low output of the event-based camera to lower the computational cost.

Creating a Layer and Learning Prototypes

An iterative online clustering method is used to learn the base patterns (hereinafter called prototypes), as it allows to process events as they are received, in an event-based manner. A layer is composed of a set of N prototypes, which all share the same radius R (which corresponds to the neuron's receptive field) and the same time-constant τ. The triplet (N, R, τ) defines a layer. First, a set of N time-surface prototypes C_i, with $i \in [\![0, N-1]\!]$, is created. The C_i are initialized by using random time-surfaces obtained from the stream of events. For each incoming event e_k we compute its associated time-surface S_k of radius R and time-constant τ. Using the L2 Euclidean distance, we compute the closest matching prototype C_i in the layer, which we update with S_k using the following rule, improved from [7]:

$$C_i \leftarrow C_i + \alpha_i \frac{S_k \cdot C_i}{\|S_k\| \, \|C_i\|} (S_k - C_i) \tag{28}$$

with α_i the current learning rate of C_i defined as:

$$\alpha_i = \frac{1}{1 + A_i}$$

where A_i is the number of time-surfaces which have already been assigned to C_i. If a prototype C_i is poorly triggered, it is re-initialized and forced to learn a new pattern. This prevents badly initialized prototypes to stay unused, and helps them converge to meaningful representations.

Building the Hierarchy

One can then stack layers in a hierarchical manner, in order to form a network (see Fig. 24). First, the visual stimulus is presented to the event-based camera (Fig. 24A), which outputs a stream of visual events. A given event e_m of the stream must go through all the layers before the next event e_{m+1} is processed. At each layer (N, R, τ), if the time-context T_m of the event e_m satisfies Eq. (27), the corresponding time-surface S_m is computed (see Fig. 24B). Then, the best matching prototype C_c is updated using Eq. (28) (see Fig. 24B). At this point, the polarity p_m of e_m is modified so that $p_m = c$, c being the ID of the best matching prototype. Event e_m is then sent to the next layer to be processed in a similar manner. We must emphasize that the first layer, which receives *visual events* from the camera does not take the polarity (that corresponds to the increase or decrease in contrast) into account for the reason exposed in section **??**. All *visual events* have their polarity p set to zero. In the subsequent layers, however, the polarity now encodes a pattern, and we refer to them as *pattern events* instead of *visual events* for which the polarity corresponds to a luminance change. Pattern events are then fed to the next layer, and processed in a similar manner. As we go higher in the hierarchy of layers, subsequent layers combine patterns from previous layers, thus their prototypes (and so the corresponding polarities) encode more and more sophisticated patterns. As an illustration, the first layer can only encode the shape and the direction of the motion. The second layer however, because it is working with the first layer output can encode changes of direction in the motion. Once the full hierarchy has been trained, meaning that its time-surface prototypes have converged, the learning is disabled: prototypes are no longer updated using Eq. (28).

The network can now serve as a feature extractor: the polarities of events output by the network will be used as features for classification. Because this algorithm is truly event-based and data-driven the computation time directly depends on the number of events transmitted by the camera.

3.3 A NEW NEUROMORPHIC DATASET: NAVGESTURE

As mentioned in the previous section, existing gesture and action recognition datasets are recorded using a non-moving camera set in front of a static background [13, 124, 162, 14, 159]. In some other popular neuromorphic datasets such as N-MNIST and N-Caltech101 [121], the event-based camera

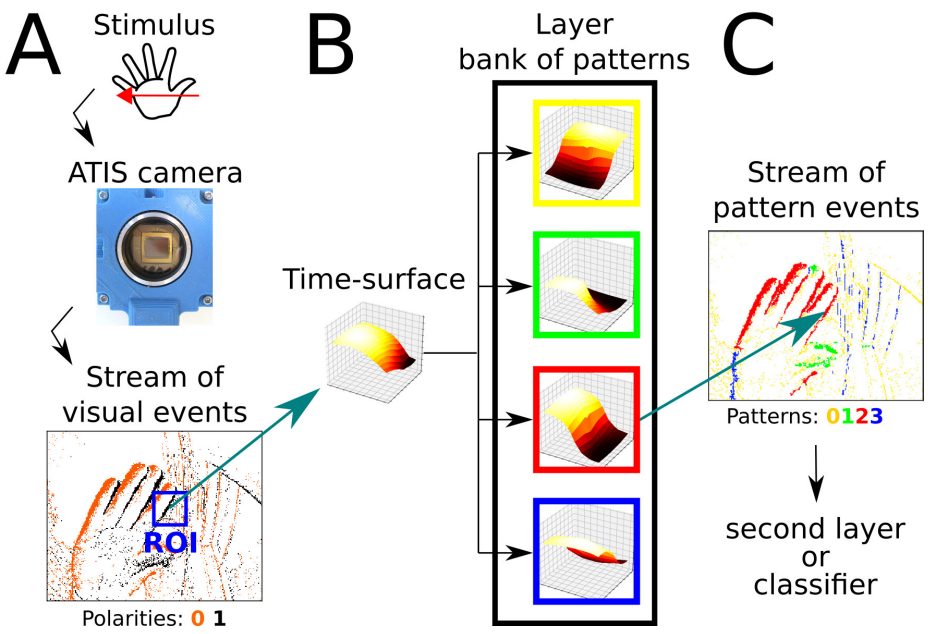


Figure 24: (A) A stimulus is presented in front of a neuromorphic camera, which encodes it as a stream of event. (B) A time-surface can be extracted from this stream. (C) This time-surface is matched against known pattern, which are also time-surfaces, and that can be used as features for classification.

is set up on a pan-tilt in front of a computer screen, hence the dynamics of recorded objects correspond to the pan-tilt movement. The same issue arises in N-Cars [123] because of the very short duration of each clip. Furthermore cars are cropped, removing most of the background.
The proposed dataset offers a challenging gesture recognition task because of its dynamic and changing backgrounds. All gestures were recorded in selfie mode, with the users holding the camera with one hand and performing the gesture with their free hand. The fact that users where holding the device leads to a wide variety of centring and gesture distance to the camera. The dataset features both right-handed and left-handed users. The users were either sitting or walking, indoors and outdoors, in uncontrolled lighting conditions. The neuromorphic camera used is an ATIS [23] with a lens VM-6.5-IR-CCD from Universe Optics. This choice was made in order to facilitate the "auto"-centring by the end-users, by allowing a larger field of view.

The NavGesture dataset has originally been designed to facilitate the use of a smartphone by the elderly and the visually impaired. The gesture dictionary has 6 gestures in order to be easily memorized. They have been selected to be the most compact set able to operate a mobile phone. Four of them are "sweeping" gestures: *Right*, *Left*, *Up*, *Down*. These are designed to navigate through the items in a menu. The *Home* gesture, a "hello"-waving hand, can be used to go back to the main menu, or to obtain help. Lastly, the *select* gesture, executed only using fingers, closing them as a claw in front of

the device, and then reopening them, is used to select an item.
The NavGesture dataset is split into two subsets, depending on whether users were sitting or walking: NavGesture-sit and NavGesture-walk. The NavGesture-sit dataset features 28 subjects, 12 being visually impaired subjects, with a condition ranging from 1 to 4/5 on the WHO blindness scale and 16 being people from the laboratory. The gestures were recorded in real use condition, with the subject sitting and holding the smartphone in one hand while performing the gesture with their other hand. Some of the subjects were shown video-clips of the gestures to perform, while others had only an audio description of the gesture. This inferred some very noticeable differences in the way each subject performed the proposed gestures, in terms of hand shape, trajectory, motion and angle but also in terms of the camera pose. Each subject performed 10 repetitions of the 6 gestures. In a second stage, all the acquired clips were manually labelled and segmented. We removed problematic clips, such as wrongly executed gestures or gestures executed too close to the camera. The manually curated dataset contains a total of 1,342 clips.
In the NavGesture-walk the users walked through an urban environment, both indoors in the laboratory, and outdoors in the nearby crowded streets in the center of Paris. Users recorded the gestures while walking, holding the smartphone with one hand and performing the gestures with the other. This uncontrolled setting leads to much more variation in pose, unwanted camera movements, dynamic backgrounds and lighting conditions. This dataset features 10 people from the laboratory that performed 5 times each of the 6 gestures. The dataset contains a total of 339 clips. An overview is presented in table 2. An example of the the "Swipe Up" gesture is shown in Fig. 22. The NavGesture dataset is publicly available at https://www.neuromorphic-vision.com/public/downloads/navgesture/.

Dataset	#users	#classes	#clips	Camera	Background	Centring
DvsGesture	29	10 + 1	1,342 + 122	Static	None	Upper body
NavGesture-sit	28	6	1,342	Handheld	Moderate	Selfie, sitting
NavGesture-walk	10	6	339	Handheld	Important	Selfie, walking

Table 2: Characteristics of the three Gesture Datasets used in this work.

3.4 EXPERIMENTS AND RESULTS

The first experiment on the Faces dataset focuses on extracting static properties. We show that a single layer is sufficient in this case. The second part is focused on gesture recognition tasks. In these experiments, the dynamic properties of gestures are paramount, and more layers allowed for better recognition scores.

Because the neuromorphic camera detects change in contrast, these can either be ON or OFF events depending on the contrast between the foreground and the background. Indeed, the same moving object could generate ON events in front of a dark background, and OFF events in front of a lighter background. This is the reason why in all the following experiments we did not take the ON/OFF polarity of visual events into account, as the polarity

is context-dependent. An example of this phenomena is a moving hand in front of a black and white stripped background, that would generate both ON and OFF events depending on the background strip. More generally, this shows that the polarity is context-dependant and can not be taken into account as valuable information per se except in the case of a controlled environment and stimulus. An example of such a controlled stimulus where the polarity can be useful is an eye-tracking task, as the pupil is darker than the iris, which is also darker than the white of the eyeball.

For all classification tasks, the output of end-layers (larger time scale) is integrated over time to generate a histogram of activity per feature as in [7]. This histogram is then used as a dynamic signature of the observed stimulus. This signature is fed to a classifier, in this case a nearest neighbor. More sophisticated classifiers could be used, but this demonstrates that extracted features are discriminative enough for classification.

3.4.1 Static properties: Experiments on the Faces dataset

This dataset contains clips of the faces of 7 subjects. Each subject was recorded 24 times, resulting in 168 clips. The subjects had to move their head in a square-shaped trajectory, by following a dot on a computer screen. The dynamic is therefore the same for all subjects, and does not carry any meaningful information for the classification task. Experiments were performed on a standard desktop computer. We performed 10-fold cross-validation with 5 examples in the train subset, and 19 in the test subset. We used a single-layer with N = 32 prototypes, receptive fields of radius R = 6 and τ = 5 ms, we obtained 96.6% recognition score on this dataset. By increasing the number of prototypes to N = 64, we achieved 98.5% in average recognition rate. We noticed that increasing τ higher than 5 ms was not beneficial and even decreased our classification accuracy. This is because time-surfaces encode both static properties such as shape and dynamic properties such as optical flow. A small τ will mainly encode static properties whereas a larger τ will also encode dynamic properties such as pseudo optical-flow. When we added a second layer, the recognition rate dropped. A single layer is therefore sufficient to encode static properties such as shape. The classification was made using a 1-nearest neighbor, and does not rely on advanced classification techniques.
In comparison, the HOTS model in [7] performed at 79% using a three-layer architecture, with its end-layer having N = 32 of prototypes. It must be noted that this improvement in recognition rate also comes with a faster computation because of the reduction in the size of used time-surfaces, from size 4624 in HOTS to size 169 in our work.
Classification scores depend on the number of prototypes: the more prototypes, the higher the recognition rate.

3.4.2 Dynamic properties: Experiments on the NavGesture datasets

In both NavGesture-sit and NavGesture-walk datasets, subjects hold the smartphone in their hand, which results in camera movements and unwanted jitters that generate background activity. In the case of the NavGesture-walk the visual background is even more present as subjects are walking while performing the gestures. The experiments were performed on a standard desktop computer, and we used k-fold cross-validation, with k the number of subjects.
In order to remove events generated by the background we used the Dynamic Background Suppression method introduced in Section 3.2.1. The DBS uses the following parameters, set experimentally:

- $\tau_b = 300\mu s$
- $\alpha = 2$
- $A_T = 5$
- grid size : 3×3

Figure 22 illustrates the effect of the DBS. Table 3 reports the mean percentage of remaining events for each gesture after removing the background. The DBS allows to remove around 40% of events before the feature extraction. This has a direct impact on processing time as we compute event by event.

Gesture	Mean number of event	Mean percentage left after the DBS
Down	988,901	41%
Home	2,398,850	48%
Left	969,014	42%
Right	962,501	43%
Select	1,212,222	30%
Up	1,110,652	44%

Table 3: Mean percentage of events left after each the Dynamic Background Suppression for each gesture class.

In our experiments we used networks composed of 1 to 3 layers. We observed that two-layers networks perform better. Some gestures such as "Select" or "Home" have changes in direction, which can be encoded by networks with two or more layers. However, we suspect that three-layers networks encode features that are too complex for the stimulus, resulting in less discriminative features and a lower recognition rate.
Because events are decayed over time, the value of τ must correspond to the dynamic of the stimulus [170]. If τ is too small, the extracted time-surface will encode only spatial information. If τ is too large, the trail of older events will blur the shape, encoding only direction of movement. In more extreme cases with τ going to larger and larger values, the resulting time-surface will carry less and less information, as all past events will have the same weight. Of course this has also a close relation with the radius of the time-surface as

larger radii can encode longer trails of events.
This observation leads to the fact that τ should be set in regard to the radius R of the time-surface and the velocity v of the apparent motion in pixel per second:

$$\tau \approx \frac{R}{v} \tag{29}$$

We observed that a first layer with a τ value in the order of 10 ms allowed to encode both shape and direction of motion (only direction, not changes in direction). The second and end-layer has a τ value of 100 ms, in order to encode changes in the direction of motion.
A direct difficulty comes from the almost fish-eye field of view of the camera: if the mobile device is not held vertically or if the gesture is a bit off-axis, it becomes very difficult at the edges of the field of view to determine if the motion is vertical or horizontal.
Ablation study In order to assess the benefits of the DBS in obtaining better recognition rates, we compared the performance achieved with and without the DBS. Results show that DBS does improve recognition rates, increasing the score from 81.3% to 92.6% when using the NavGesture-walk dataset, as shown in table 4.

ID	Dataset	Layer 1			Layer 2			DBS	Classifier	Results
		N	R	τ	N	R	τ			
E1	NavGesture-sit	8	2	10 ms	8	2	100 ms	✓	k-NN	95.9%
E2	NavGesture-walk	8	2	10 ms	8	2	100 ms	✓	k-NN	92.6%
E3	NavGesture-walk	8	2	10 ms	8	2	100 ms		k-NN	81.3%
E4	NavGesture-walk	8	2	10 ms				✓	k-NN	88.7%

Table 4: Summary of obtained results on the NavGesture dataset. The use of the Dynamic Background Suppression in E2 allows to drastically improve the recognition rate by over 10% compared to E3. Also, the addition of a second layer is beneficial, as shown by the improvement in E2 compared to E4

3.4.3 Experiments on the DvsGesture dataset

Amir et al. [13] released a 10-class (plus a rejection class with random gestures) dataset of hand and arm gestures, performed by 29 subjects under 3 different lighting conditions. The camera is mounted on a stand while the subjects stood still in front of it. This dataset has no background so the DBS was not used. Authors split the dataset into a training set of 23 subjects and a testing set of 6 subjects, preventing cross-validation for comparison purposes. We used the same 2-layer network architecture as the one used for NavGesture. The only difference is that we increased the number of prototypes in the last layer because the gestures are more complex. In order to take into account the spatial component of gestures, we split the pixel array into sub-regions, using a 3×3 grid. This is possible because the centring is very similar for all clips in the dataset. Hence, the final feature is a histogram of size $3 \times 3 \times 64 = 576$. We achieved a classification accuracy of 96.59% for the 10-class subset and 90.62% for the 10 classes plus the rejection class. One can observe in the confusion matrix (Fig. 25) that "Hand clap", "Arm roll",

"Air guitar" and "Air drum" are the only gestures that are mistaken. These gestures all share very similar hand movements at the same spatial location, located in front of the torso. "Arm roll" and "Air drum" are also very similar. Their difference lie in the fact that hands in "Arm roll" move along the same vertical line, and we suspect that the receptive field is too small to capture this information.

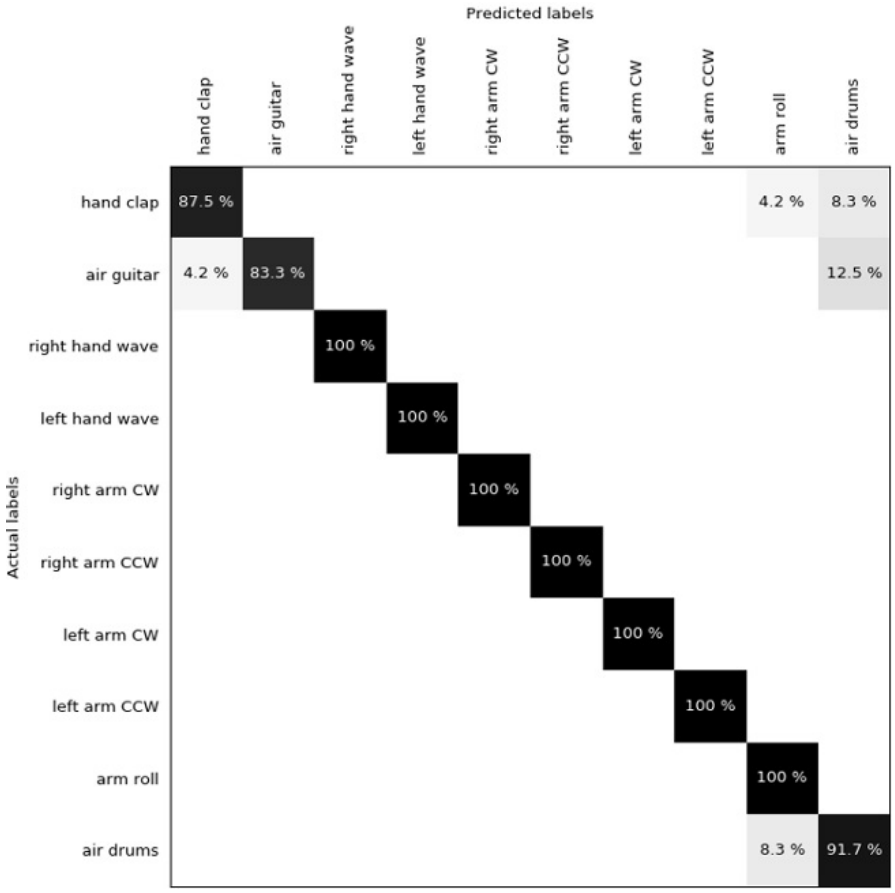


Figure 25: Confusion matrix for DvsGesture using 10 classes. Global accuracy is 96.59%. "Hand clap", "Arm roll", "Air guitar" and "Air drum" are the only gestures that get confused. The reason might be that they generate similar motion in the same spatial location.

When adding the rejection class, the same gestures get confused (Fig. 26). Indeed, only one clip of "Left hand wave" gets mistaken for "Air guitar", which is understandable as the left hand in these two classes performs the same movement at the same location. The global accuracy decreases mostly because of the "Hand clap" that gets misclassified more often and because of the "Other gestures" that also are harder to classify.

One can observe in Table 5 that for the 10-class classification task our system performs in the same range of accuracy using a k-NN as other very elaborate systems using state-of-the-art neural networks.

It must be noted that the same time constants gave best results for both NavGesture and DvsGesture, which shows that decay must be chosen in accordance with the stimulus, in both case gestures. Indeed, previous works such as HOTS [7] and HATS [123] used decay times that were three orders

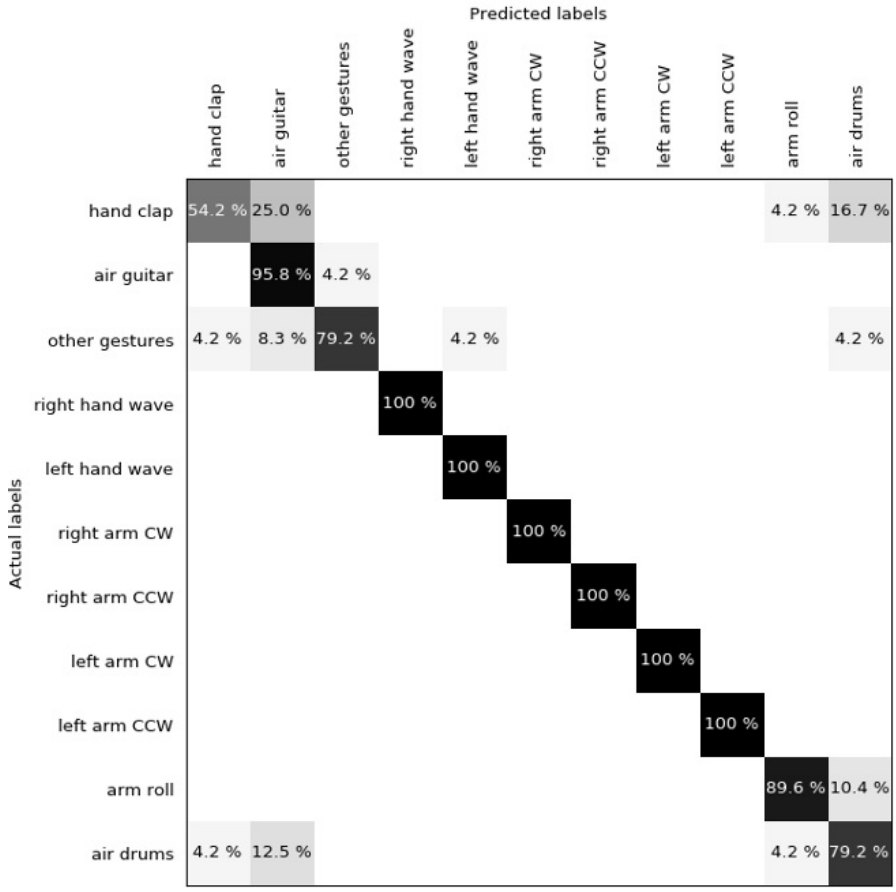


Figure 26: Introducing the rejection class "Other gestures" amplifies the mismatch between the four precedent gestures, leading to a global accuracy of 90.62%. However, it has almost no impact on other gestures (4.2% in the "Other gestures" row corresponds to only one clip).

of magnitude higher than the duration of the stimulus. This resulted in time-surfaces that acted as binary frames instead of encoding the dynamics of the scene. Furthermore, such high decay values resulted in the incapacity of forgetting past events.

3.5 IMPLEMENTATION ON A SMARTPHONE

The proposed gesture recognition pipeline has been implemented on a smartphone [120], a Samsung Galaxy S6 (model GM-920F), with a custom Android application allowing easy navigation through basic smartphone functions, such as making a call or sending a pre-defined text message (see Fig. 27). The event-based camera was directly plugged into the micro-USB port of the smartphone (see Fig. 20). The gesture recognition module is implemented in native C++ using JNI to communicate with the Android application.

The gesture recognition module consists of basic noise filtering (a refractory period followed by a spatio-temporal denoiser, known as the *background activity filter*, that removes pixel electrical noise), the Dynamic Background Suppression, a 1-layer Feature Extractor ($N = 8$, $R = 2$, $\tau = 10$ ms,) and a

	Method	DvsGesture (10 classes)	DvsGesture (10 classes + 1)
Amir et al. [13]	CNN (avg 192ms)	91.77% (96.49%)	91.77% (94.59%)
Shrestha et al. [172]	SLAYER		93.64%
Kaiser et al. [173]	DECOLLE		94.18%
Ghosh et al. [14]	ST filter + CNN (avg 200ms)		94.85% (95.94%)
Kaiser et al. [183]	SNN eRBP		92.7%
Wang et al. [174]	PointNet++ (avg 118ms)	96.34% (97.08%)	94.10% (95.32%)
This work	Time-surfaces + k-NN	96.59%	90.62%

Table 5: Comparison in accuracy of state-of-the-art methods for the DvsGesture dataset. When noted (avg) an averaging scheme was proposed to improve the system accuracy. Our method, although using a simple k-NN classifier performs in the same range for the 10-class classification. However, the k-NN lacks the discriminative power to handle the rejection class on the contrary of more sophisticated classifiers.

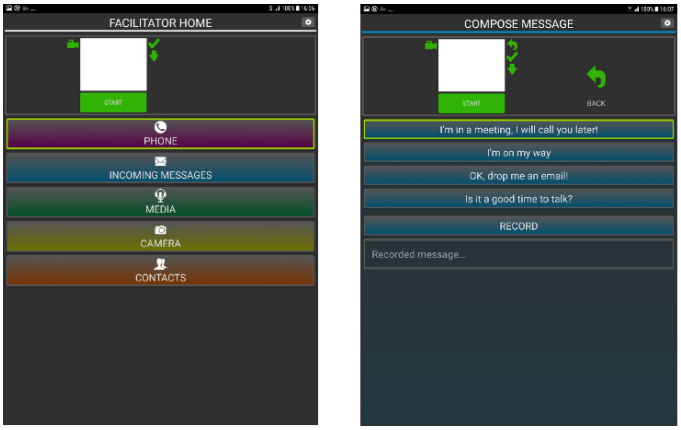


Figure 27: Interface of the Android application that was developped in order to operate the phone using the proposed gestures. Right is the main menu, left illustrates the pre-defined messages the user could send.

k-NN classifier.
We used two strategies to segment gestures, the first one is an "auto-start" based on the global visual scene activity. This option works when users are seated but is inadequate for walking cases. The second strategy relied on pressing a button before a gesture to start the recording. The duration of the recording was tuned experimentally to 2 seconds which seems to be the experimental upper bound of the duration of a gesture. This two-second batch of events at once to the gesture recognition module, that returns the gesture class to the Android application to be converted to an Android command. An overview of the system is presented in Fig. 28.

To assess processing time, we ran five trials for each gesture in two different settings. The input event stream having a duration of 2 seconds, a real-time processing is reached when the processing time is below 2 seconds. In the first scenario, the smartphone was set on a table. In the second scenario the smartphone was handheld in selfie mode, with the user walking around. All results are compiled in Table 6. When looking at the first scenario, we can see that all gestures are under the 2 seconds barrier, except for the "Home" gesture (a "Hello-waving" gesture). This is because this ges-

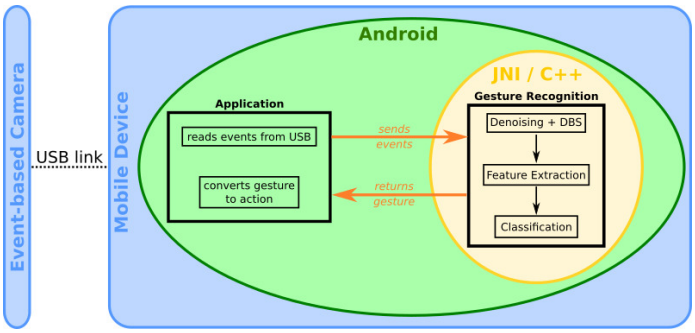


Figure 28: Overview of the Android smartphone system.

ture produces 3 times more events than all other gestures (see Table 3). The algorithm being truly event-based, the processing time directly depends on the number of events to process. Also during trials 3 and 4, the user waved his hand 5-6 times, while in trials 1, 2, and 5 waved only 3-4 times. The second scenario is the handheld selfie mode scenario, where the background generates a high number of events, hence necessitating longer processing time. However, all gestures except for the gesture "Home" that could be computed in real-time. This gesture should be replaced by another more event-based friendly gesture that would generate less events, or should be more constrained by forcing users to only wave 1 or 2 times.

This prototype was tested by untrained visually impaired end-users, in real use conditions. The subjects were asked to perform certain tasks to operate the smartphone. These preliminary tests lead to a global accuracy of 78%, which is below the 88.7% accuracy we obtained using the same single layer on the NavGesture-walk dataset. We suspect this is partly due to framing and off-axis handling of the smartphone.

3.6 DISCUSSION AND CONCLUSION

We introduced in this chapter a proof of concept for an event-based Android application for gesture recognition using the computing power of a mobile phone. The main idea was to show that it is possible to make full use of the high temporal resolution of event-based cameras on a power-constrained device. The system used a camera designed to operate with Android using the USB link to stream events. This is by far a very inefficient way to input data to the mobile platform as USB is often too slow and implies time stamping events that adds more bits of information to the acquired events. It is expected that if this type of camera is one day introduced in a mobile device it will use better connectivity such as MIPI buses which are designed for low-power applications and eventually an associated processor. This will remove the need for time stamping and allow both direct routing to the processor and direct computation on the time of arrival of events with no delays. Due to the limitations of the developed software we used 2-second packets of events to optimize communication within the smartphone. However, we showed that processing required in most cases

Processing time in ms for 2000 ms of input Setting: fixed position (no background)						
trial	Up	Home	Right	Left	Select	Down
1	132	2343	54	127	40	54
2	57	2798	60	56	57	45
3	74	3047	44	275	61	42
4	254	3833	32	42	29	54
5	48	2107	28	45	47	51

Processing time in ms for 2000 ms of input Setting: outdoor – moving						
trial	Up	Home	Right	Left	Select	Down
1	320	4119	154	641	138	115
2	614	3669	704	282	265	451
3	468	4305	854	421	551	342
4	569	3681	575	548	956	371
5	899	3890	722	354	892	620

Table 6: Processing time in milliseconds for five trials of each gesture on the mobile phone, depending on two conditions. ”Fixed position” corresponds to a mobile phone set on a table, which means no background. ”Outdoor, moving” corresponds to handheld selfie mode, while walking around. Each gesture corresponds to 2000 ms of events, meaning that except for the ”Home” gesture, all proposed gestures can be processed on real-time. The event-based camera is data-driven so a gesture like ”Home” which corresponds to several ”swipe” gestures will generate more events (see Table 2). Our algorithm being truly event-based it is also dependent on the number of events, and takes more processing time the more events it receives.

less than 2 seconds per batch, which implies that real time performance can be reached if transmission delays are solved. We are confident that a way can be found within Android to transmit events from the camera to the processing stage with no latency. We have also shown that it is possible to handle the stream of events in an asynchronous manner. This allows the temporal machine learning algorithm to be efficient while using only a single core of the smartphone. The hierarchical temporal network has been optimized for the set of defined gestures showing that robust recognition levels can be reached without requiring the use of GPU or using the non event-based concept of generating frames from an event-based sensor. Experimental results show that as expected the computation is scene dependent and therefore tightly linked to the amount of events generated by the observed object.

We have also shown that the temporal precision of event-based cameras can tackle different tasks, where it would have been too computationally expensive or even impossible to compute with frames. As an example, the background suppression algorithm that for the first time considers outdoor, hand-held scenarios relies on the simple idea that the foreground being closer to the camera will on average generate more events than the background. The idea of using the relative mean activity for background suppression shows that high temporal precision is a valuable feature as it implies that velocity is linked to the amount of data produced, and can be estimated precisely. Moreover, the use of well designed temporal filters can reduce even more the already sparse stream of events, leading to faster event-by-event computation. There is still much to develop around the concept of using time as a computational feature. As an example the use of scene dynamics allows to derive techniques such as the one in [184] that uses the temporal signature of eye blinks to detect the presence of a face in a scene. This approach introduces an alternative to the current stream of thought that believes everything has to be learned using large databases.

All data collected and used in this work has been made available to the community. The introduction of this new database will set the groundwork for further work on dynamic background suppression.

4 THE NEED OF INCREMENTAL COMPUTATION AND TIME ADAPTIVE RESOURCES MANAGEMENT FOR EVENT-BASED SENSORS

The properties of event-based sensing provide a powerful way to overcome classical limitations of traditional image-based perception such as motion blur and the trade-off in selecting frame rates: losing dynamic information at lower frame rates, or setting a high frequency acquisition rate leading to prohibitive computational costs. As event-based sensing has revolutionized the way visual signals are acquired, the algorithms and techniques to process events are also expected to undergo the same changes. In this chapter, we show why applying conventional image processing techniques to events - i.e. transforming them into frames - is a fundamentally wrong approach, as it eliminates the essential properties, and therefore advantages, of using event-based cameras in machine vision. This work provides a new generic architecture that allows for incremental processing and a time adaptive resource management relying on associative in-memory computation that are consistent with the data driven flow of events. Although shown in an event-based machine learning context, the approach applies to all machine vision applications. We also provide a study of a practical hardware implementation of the proposed architecture showing the advantages of processing event-by-event as they are output by the sensor rather than recycling inadequate techniques from conventional image-based methodology.

4.1 INTRODUCTION

Neuromorphic event-based sensing and computing are parts of an effort undertaken thirty years ago in a quest to move towards computation inspired by biological systems. As phrased by C. Mead in [185]: "Biological information-processing systems operate on completely different principles from those with which most engineers are familiar. For many problems, particularly those in which the input data are ill-conditioned and the computation can be specified in a relative manner [...]." Relative information acquisition allows neuromorphic event-based sensors to retrieve information "continuously" so that what has been already sensed, transmitted, and processed does not need to undergo the same process again, saving energy and computational resources on behalf of the system [186]. This paradigm differs from conventional computer vision that relies on the use of stroboscopic frames where information is recorded regardless of the dynamic content present in the scene.

The use of frames is at the core of all developments in computer vision since its inception and has led to numerous techniques based on these

dense, redundant, and synchronous representations including recent machine learning hype such as Convolutional Neural Networks (CNNs) which have become very successful for image classification since the seminal work in [187]. It is therefore tempting to reuse concepts from conventional machine vision when considering event-based data because of the wide availability of methods and dedicated hardware like the GPU. However, as we will show, these are inadequate when dealing with the high temporal resolution data output by event-based cameras. Recycling methods that have been designed to operate on images where time is generally disregarded can hardly make full use of these sensors and will lead to a large waste of resources in terms of both memory and computation. Generating images from events is not necessary because it is possible to approach the same problems by considering the timing of events and developing incremental techniques where each event adds to what has been computed [188].

Another major limitation of using frame based techniques on event based data is the memory bottleneck, which is the main limitation in current CNNs and spiking neural networks accelerators [189][190]. The large number of memory accesses needed to deploy image based algorithms tends to increase the system latency. In addition, implemented memories in these accelerators usually store all the spatial resolution of the sensor as for image-like representations. A large portion of the allocated memory when considering the sparsity of the output of neuromorphic vision sensors is not used and hence wasted while requiring memory access. Existing work such as [191] tries to reduce the memory access limitations by updating the rows and columns of the memory for each incoming event. In [192], sparsity is taken into account to save memory on normalized frames constructed from DVS events to execute frame-based CNNs thereby reducing the memory bottleneck. Perhaps with the advancements in memristor technology, there will be an improvement in memory accesses but the technology hasn't currently matured [193].

In the following sections, we show that when considering the native sparseness properties of event based acquisition and the adequate processing by operating incrementally on each incoming event, it is possible to develop a more adapted and efficient processing while being efficient in both memory and computation requirements. We review existing and widely used available event-based dataset, and show that a dynamic memory allocation requires a much smaller memory footprint than image-like representation while being as efficient. We also introduce the concept of a generic memory architecture that allows for improved memory access and latency (tested on FPGA). Although the ideas and concepts can be applied to any form of event based computation, we considered, as a test-bed, deep temporal learning networks as introduced in [194] specifically in the context of a gesture recognition task to comparison with frame-based neuromophic architectures [13].

4.2 THE NEED OF INCREMENTAL PROCESSING OF EVENTS

Data acquisition by the event-based sensor is driven by the scene's dynamics. Fig.29 shows the number of events generated by an event-based

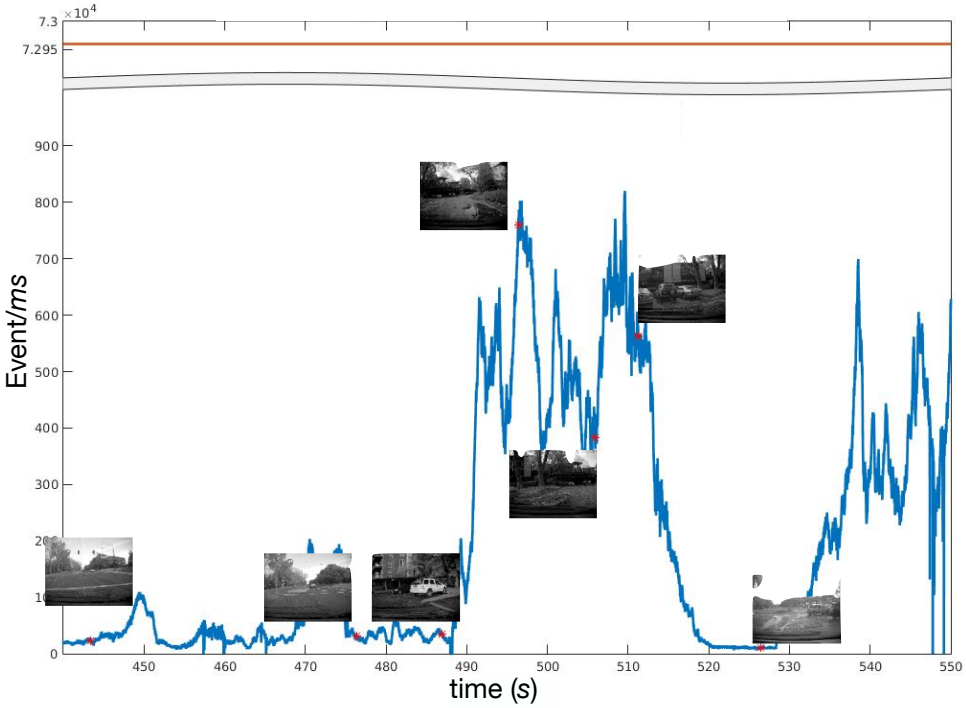


Figure 29: The "amount of data" acquired by an event-based camera mounted on a moving vehicle and its frame-based equivalent in order to achieve similar temporal precision while keeping the same spatial resolution. (BLUE) Number of events generated by the event-based camera with a temporal precision of around 1 ms. Because the acquisition is scene-driven, the number of events fluctuates over time. (RED) Number of acquired pixels for the corresponding (simulated) conventional camera that would run at 1 kHz.

camera mounted on a moving vehicle (shown in blue) *versus* the number of acquired pixels using a simulated conventional camera that would offer the same temporal precision, while keeping the same spatial resolution. Because the event-based camera has a temporal precision of around 1 ms, its frame-based equivalent should run at 1 kHz. Even in the case of a moving camera, which can be seen as an adverse situation for event-based cameras, the event-based sensor outputs around 90% less "data" than its frame-based equivalent for the same visual scene. However it must be noted that event-based sensors output the precise *timing* of changes in the visual scene through timestamps, while conventional cameras output *absolute light intensity*.

Because frame-based cameras perform a global update of the visual scene, current vision algorithms operate on dense image-like representation, as the complete spatial resolution is stored. This guarantees that all the information for all pixels is available, but this also makes the assumption that the information in every pixel is valuable. Furthermore, it means that all pixels must be processed at each clock tick, even if their values have not changed.

On the contrary, event-based sensors output sparse streams of events, each event being a local update of the visual scene. It is hence more logical to operate directly on each new incoming event, in order to update a model or a representation. Therefore, to make full use of these sensors, it is important

to design local and iterative algorithms. An event-based algorithm must operate directly on each incoming event to update what is being computed rather than creating frames by accumulating events. Generating frames by accumulating events will not only increase latency, thus losing the advantage of these sensors, but also cause a massive waste of resources both in memory and computation as most generated frames from the event-based frames will be redundant since changes happen only for a small population of pixels.

In numerous event-based algorithms [195, 27, 28, 25, 29], only an event's spatio-temporal neighborhood is needed to compute a local feature and/or update a model. Some algorithms also make the assumption that information carried by events loses importance over time [13, 196, 31], by applying temporal decays. Hence, only the most recent events are to be stored instead of storing the last value at each and every spatial location like in images. It is then common sense to only store recent events that carry valuable information as a dynamic list. This will prevent older events from being re-processed and more importantly, allows for a smaller memory footprint. This memory requirement is similar to the conventional cache memory in a general purpose computer, as it stores only recently accessed data.

4.3 THE USE OF FRAMES IN NEUROMORPHIC EVENT-BASED VISUAL PROCESSING

In order to take advantage of decades of frame-based computer vision, numerous methods were proposed to convert events into frames [9, 10, 156], sometimes with the help of a CNN [11, 12]. In order to cope with blurred frames obtained with conventional cameras, some authors proposed methods to use events in order to de-blur these frames [197, 198]. Once frames are obtained from events, or frames are de-blurred using events, one can easily use a CNN for different tasks, such as optical flow computation [15, 16], object recognition [17, 18, 19, 20, 14], SNN training [199] or even denoising [21]. Video and image reconstruction is a very active topic in event-based machine vision. Several techniques are being used to generate images from events. In [200] an Extended Kalman Filter (EKF) is used to reconstruct images using events. While in [201], a sliding spatiotemporal window of events is used to estimate grey levels and optical flow. In [202], the authors generated video streams from events using high-pass convolutions. In [203] a manifold regularization of events integrated over time into a surface of active events [196] to reconstruct video from events. In [204] greedy CNNs are applied to generate grey levels from events. Maqueda et al [205] use accumulated events over 50ms time windows as input to a CNN running on a GPU to estimate the steering angle of the wheel of a car.

4.4 CNNS IN EVENT BASED VISUAL PROCESSING

Convolutional Neural Networks (CNNs) are a widespread approach in image classification. They represent perhaps the most inefficient approach for

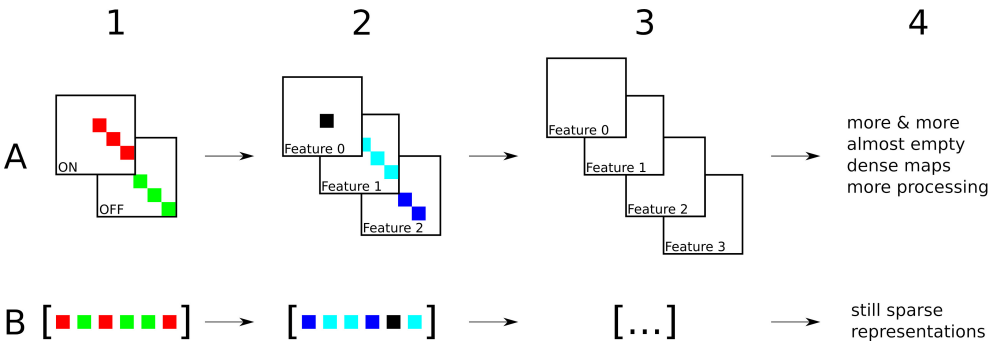


Figure 30: (A) Storing events into dense image-like representations like features maps, like in CNNs. (B) Instead it is possible to store events into a list in order to preserve the sparseness of the stream of events.

reconditioning events into frames due to their large memory requirements. Despite having been designed to operate on images, CNNs are now widely used to process the sparse and asynchronous stream of events output by event-based sensors [206][13]. This approach has led to events-to-frame-to-events conversions such as in [155]. Using CNNs or any other frame-based method to process stream of events is indeed taking advantage of decades of research in frame-based computer vision, but it comes at the cost of loosing the sparseness and low-redundancy of event-based sensor. Indeed taking the best of both worlds and using a combination of frames and events has resulted in interesting research, seen in [207][208][192]. A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers consist of a series of convolutional layers that convolve the input with a kernel using multiplications or other dot products. The convolution layers create more abstract feature maps and require more and more memory allocation as the network gets deeper as shown in Fig. 30(A). Incoming events from an event based sensor shown in green and red in the first layer of the CNN in Fig.30(A).1 are sparse, therefore most of the allocated feature planes are under-used. As the network gets deeper and the number of kernels increases, (Fig.30(A).2-4) the number of unused memory allocations will also dramatically increase.
Beyond memory loss, using frame-based algorithms on purely event-based data for machine learning results in several drawbacks:

- it re-introduces redundancy when frames are built from temporally overlapping batches of events;
- generating frames from events is computationally expensive and contradicts the founding principles of energy efficiency of neuromorphic hardware;
- the increase of latency and/or the loss of temporal accuracy by binning batches of events;
- the spatially dense representation that is traded for when events are turned into frames, resulting in more convolutions than needed as will be shown;

- the memory use is increased as the dense image matrix representation must be stored instead of a sparse stream of events, as we will also show.

A sensible approach is to store incoming events in compact dynamic structures as shown in Fig. 30(B). List based storage is surely the most adequate method to store this type of sparse information. This matches the requirements of numerous event-based algorithms [195, 27, 28, 25, 29] where only an event's spatio-temporal neighborhood is needed to compute a local feature and/or update a model. It also matches another class of algorithms that rely on the assumption that information carried by events loose importance over time [13, 196, 31] by applying temporal decays.
A chained list memory type allows to adapt the constant changing number of valuable events as it has no predefined size. Instead, two criteria can be used to remove older events:

- a temporal criterion, noted T, used to remove older events. Events with a timestamp $t \mid t \leqslant t_{now} - T$ are purged from the memory;
- event uniqueness: if two events have the same address $\boldsymbol{x}$ and polarity p, only the most recent event is kept.

Events can be stored as they are received, with ascending timestamps. To remove older events that do not meet the required temporal criteria, the memory simply needs to be shifted, by updating the memory pointer to the next available memory address. This is an iterative process where older events are removed until the oldest event meets the temporal criteria. Perhaps the most adequate practical approach is to both dynamically allocate resources and access fast local memory using content-addressable memory (CAM) also known as associative memory or associative storage [209]. These are particularly adapted to the Address Event Representation format used by event based sensors [210]. Local neighborhoods can be extracted in one or two clock cycles by comparing input search data against a table of stored data, that then returns the address of matching data. This approach not only matches the requirements of event-based high temporal precision and fast processing requirements but it also solves for time consuming memory search specially when dealing with sparse data.

4.5 TEMPORAL DYNAMICS AND DATA LOAD IN EXISTING EVENT-BASED DATABASES

We analyse the content of four event-based datasets that provide different stimuli and spatial resolutions. The scope is to compare the memory footprint based on the content of these databases when considering a static frame based allocation vs a dynamic scene-driven approach. Three of the used databases (PokerDVS [211], N-MNIST [121], and DvsGesture [13]) are widely used as benchmark references by the neuromorphic community. The fourth, NavGesture, was recently introduced in [194].
The details of each database are:

- 'PokerDVS' features cropped poker card pips displayed at very high-speed in front of the camera. Each clip is 5 to 10 ms long. Pips are cropped to a 35×35 pixel array. The Mean Event Rate (MER) is 170.4 kev/s (kilo-events per second), which results in an Individual Pixel Mean Firing Rate (IPMFR) of 138 kev/s, the highest of all presented datasets, two orders of magnitude higher than non-cropped datasets;
- 'N-MNIST' features 0-9 digits with a sensor size of 28×28. Digits are acquired using a moving event-based camera in front of a computer screen displaying the original MNIST dataset as explained in [121]. N-MNIST performs 3 small displacements of 2-3 pixels, with a pause of 100 ms between each movement. Each clip has a duration of around 300 ms. The dataset has a MER of 13.6 kev/s, the lowest of all four datasets. The IPMFR is 17 kev/s;
- 'DvsGesture' features hand gestures recorded using a fixed DVS, centered at the upper body in front of a static background. Hence, it features no background. The sensor array size is 128×128. In most sequences, the upper body is almost static and generates very few events; only the arms and hands are usually moving and therefore visible. It means that most pixels at the periphery of the sensor are mostly inactive. Gestures include hand claps, arm rolls, air guitar etc. Its MER is 56.9 kev/s, and the IPMFR is 3.4 kev/s;
- 'NavGestures-walk' features hand gestures designed to control a smartphone embedding an event-based sensor. Gestures are recorded in selfie-mode, while walking in an urban environment. Hence, the database contains significant background clutter and pixel activity when compared to DvsGesture. Gestures include sweeps, hand waving etc. The pixel array is the larger of all datasets, with a size of 304×240. Its MER is 188.6 kev/s, and the IPMFR is 2.6 kev/s.

For each of these datasets, we compute the number of events in a given time-window T. The values of T are set to 1 ms, 10 ms and 100 ms. The values of 1 and 10 ms represent widely used integration times to extract meaningful features from incoming events for fast and conventional scene dynamics. The 100 ms integration time is related to extreme cases to show what can be encountered when considering deep temporal networks layers, it can be seen as the extreme upper bound of memory use.

Each events is encoded using 64 bits of dynamic memory: 32 bits for timestamp, 31 bits for spatial location and 1 bit for polarity. However, it must be noted that the spatial location could be encoded with less bits for all 4 datasets as the spatial resolution ranges from 28×28 to 304×240, meaning that only 12 to 17 bits would be needed to encode the spatial location instead of 31 bits used here. This choice has been made to ease comparison between different spatial resolution of the datasets and provide a comparison in the worst case scenarios for scene-driven dynamic allocation.

The dynamic memory allocation is compared to an image-like representation of events. We compute the amount of allocated memory needed to store

32-bit timestamps of events, as the spatial location is implicitly stored in the allocated array. Since we store both events' polarities (ON and OFF), the total amount of allocated memory required for these 2 arrays (for OFF and ON events) is then given by: $S_R \times 2 \times 32$ bits, with S_R being the resolution (number of pixels) of the recording sensor in the dataset.

Results are shown in Table 7 for time-windows ranging from 1 ms to 100 ms (Line 1). The mean number of events (Line 2) is computed over the whole dataset. We also indicate the maximum number of events (Line 3) encountered over the dataset. This maximum number is hence the *worst*-case scenario for the complete dataset.

Fig. 31 shows the distribution of the size of the dynamic memory required for all time-windows of 3 datasets: PokerDVS, N-MNIST and NavGesture-walk. These distributions show that extreme cases happen very rarely. It also shows that the dynamic memory stays below the memory capacity needed for the image-like representation by several orders of magnitude. For PokerDVS, as clips have a duration from 5 to 10 ms, using a 100 ms time-window is almost similar to counting all incoming events. This explains its asymmetric shape compared to other durations. In short, the 100 ms time-window is not adapted to the dynamics of PokerDVS and should not be taken into account given its characteristics. However it should be noted that even in this case, the dynamic allocation is still below in terms of memory footprint than the static frame representation.

As shown in Table 7, the worst case for databases like NavGesture-walk (which have the highest MERs of all datasets), for a 100 ms time-window in terms of pixel activity are respectively 86% and 93% (Line 2). In these extreme cases, the memory footprint is comparable but still inferior to the image-like representation. It is important to emphasize that since sequences from PokerDVS are 5 ms long in duration, and only sometimes reach 10 ms. It hence makes no sense to integrate events for periods over 10 ms. This dataset contains cropped pips recorded during card shuffling. The density of events is the highest among all datasets. When the time-window is set in accordance with the timescale of this dataset, which is at the millisecond point, the memory footprint requirement can drop by 90% in the average case. In the worst case, it drops by 70%.

In the case of NavGesture-walk, as gestures are executed closer to the camera, they have a higher "apparent" focal plane speed. Integrating events over 100 ms (equivalent to 10 FPS) would result in the equivalent of frame-based motion blur. As shown in the previous Chapter, the proper integration time for this dataset is around 10 ms. At this timescale, the proposed dynamic memory uses 4% of the capacity needed for the image-like representation in the average case. Even in the extremely rare worst cases, it requires half of the image-like memory footprint.

The same conclusions apply to DvsGesture that exhibits similar statistics, even though it contains no background activity. N-MNIST features digits recorded using 3 quick and brief camera movements, mimicking micro-

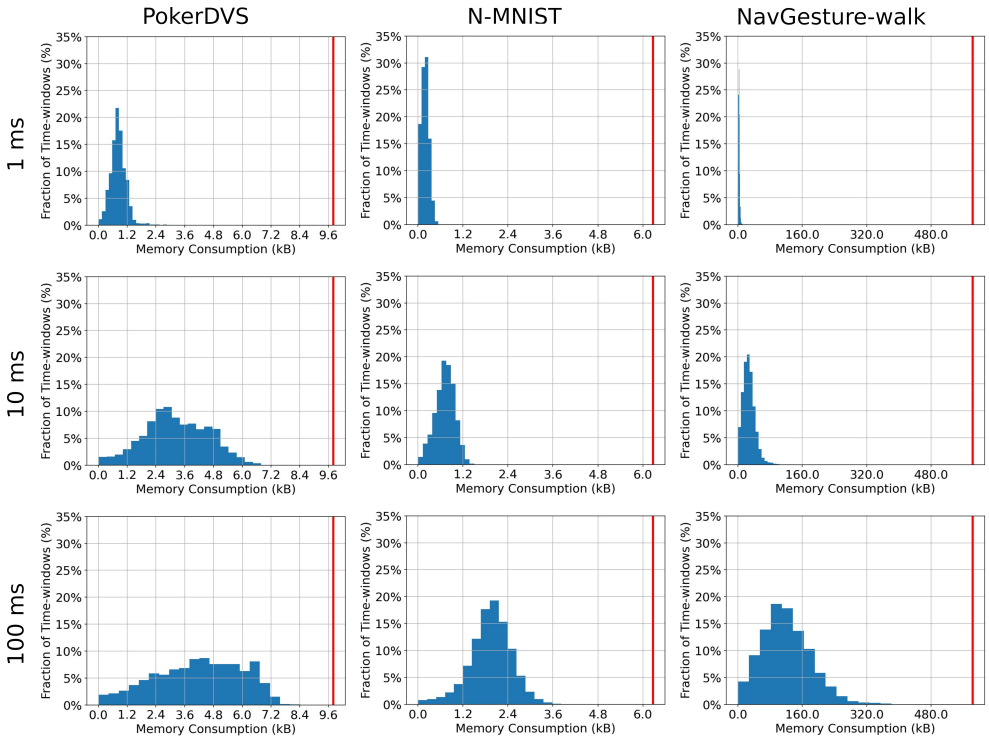


Figure 31: Distribution of the memory needed to store all events in 1, 10 and 100 milliseconds time-windows for 3 datasets: PokerDVS, N-MNIST and NavGesture-walk. The blue histogram is the distribution of time-windows in regards to their memory footprint. The red vertical line represents the allocated memory needed for one image-like representation, which is never reached when using the dynamic memory.

saccades, separated by 100 ms breaks. This means that events are generated during the camera movement, then almost no events are generated for 100 ms, before the next movement. The information needed for the classification of the digit is purely static (shape), as the dynamics are the same for all digits. This should be the perfect use-case for long integration time and image-like representation. However, the dynamic memory offers again a much reduced memory footprint than image-like static allocation for all integration times in the range of 3-11 % for most regimes at respectively 1 ms and 10 ms integration time and around 29% for upper bound cases.

These results show that for all the considered datasets -that cover a wide range of scenarios- allocating the whole sensor array in memory results in a massive waste of resources. This difference between static vs dynamic is orders of magnitudes enhanced when considering high temporal precision (1-10 ms) that correspond to the natural use-case for event-based cameras.

Figure 32 shows a practical case of previous results from the NavGesture-walk [194] dataset. It represents the memory needed to store the events over time during a hand gesture. When using an image-like representation where pixels encode timestamps, one must allocate the whole pixel array. On the other hand, using a dynamic memory with a fixed time-window leads to a memory size that adapts itself to the visual scene. In this example, the

Dataset (Mean Event Rate & Sensor Size)	**PokerDVS** 170.4 ev/ms 35x35			**N-MNIST** 13.6 ev/ms 28x28			**DvsGesture** 56.9 ev/ms 128x128			**NavGesture-walk** 188.6 ev/ms 304x240		
1. Time Window (ms)	**1**	**10**	**100**	**1**	**10**	**100**	**1**	**10**	**100**	**1**	**10**	**100**
2. Mean Number of events in TW (percentage of active pixels)	101 (8%)	390 (32%)	486 (40%)	22 (3%)	84 (11%)	229 (29%)	53 (<1%)	340 (2%)	1751 (11%)	285 (<1%)	2818 (4%)	13279 (18%)
3. Max Number of events in TW (percentage of active pixels)	356 (29%)	848 (69%)	1052 (86%)	223 (28%)	312 (40%)	597 (76%)	467 (3%)	2056 (13%)	9191 (56%)	2599 (4%)	18296 (25%)	68128 (93%)
4. Working Memory Size (kB) Dynamic - Average case	0.8	3.1	3.9	0.2	0.7	1.8	0.4	2.7	14.0	2.3	22.5	106.2
5. Working Memory Size (kB) Dynamic - Worst case	2.8	6.8	8.4	1.8	2.5	4.8	3.7	16.4	73.5	20.8	146.3	545.0
6. Allocated Memory Size (kB)	9.8	9.8	9.8	6.3	6.3	6.3	131	131	131	584	584	584
7. Memory ratio dynamic/static (Average Case)	8%	32%	40%	3%	11%	29%	1%	2%	11%	1%	4%	18%
8. Memory ratio dynamic/static (Worst Case)	29%	69%	86%	28%	40%	76%	3%	13%	56%	4%	25%	93%

Table 7: This table shows, for 4 event-based datasets, and for 3 different time-window sizes (Line 1), ranging from 1 to 100 ms, the mean (Line 2) and maximum (Line 3) number of events. Using events of size 64 bits (32 bits for timestamp + 32 bits for spatial location and polarity), it shows the needed memory for the dynamic memory (Lines 4-5) and the corresponding allocated memory needed for the image-like representation (Line 6). Static memory usage is increased relatively to dynamic memory usage by factors ranging from 2 to 100 when considering short time-windows (< 10 ms, resp. high frame-rates > 100 FPS). For time-windows of 100 ms (10 FPS equivalent), this memory need is, on average, half the capacity of a static memory, but worst-cases can go up to similar levels. (Lines 7-8). However the use of event-based sensor at such low temporal resolutions is in any case questionable.

Memory usage factors are closely linked to the time-window size, which can be linked to the frame-rate for conventional frame-based cameras. It is also dependent on the sensor array size. It shows that in order to achieve a temporal precision of 1 to 10 ms (which corresponds to frame-rates of 100 to 1000 Hz), the image-like representation requires 25 to 100 times more memory capacity. This means that it also requires faster memory bandwidth to transmit the data.

memory consumption is over one order of magnitude higher when using the image-like representation instead of the dynamic list.

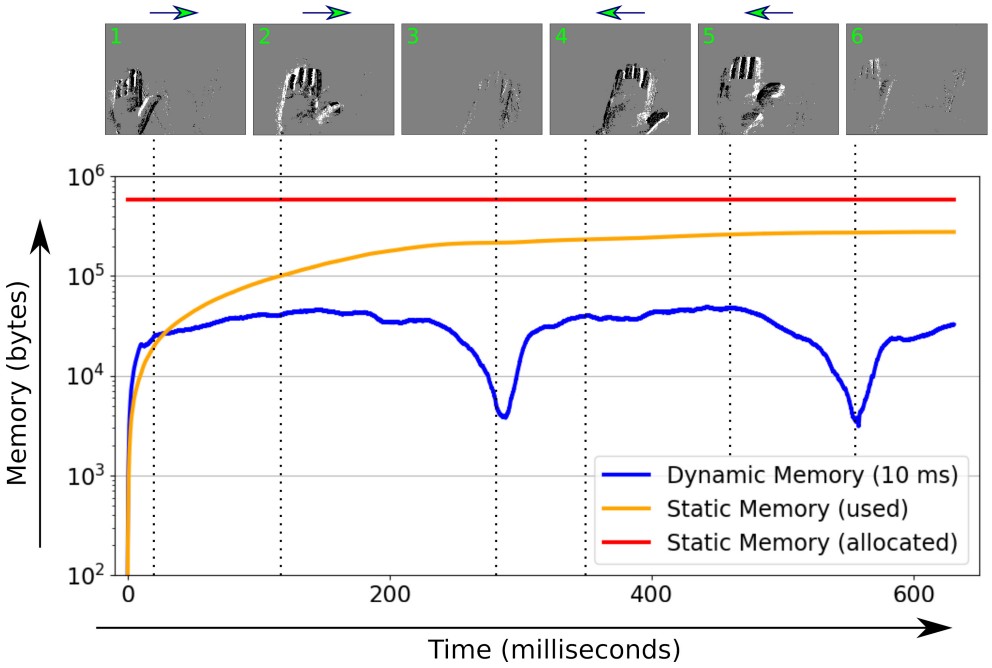


Figure 32: Comparison of memory usage when using a dynamic list memory instead of a static matrix memory when extracting time-surfaces. The red line represents the total allocated memory when using a image-like frame-based representation while the orange line represents the actual memory used. The blue line represents the memory consumption when using a dynamic, time-windowed memory. The dynamic memory has a temporal window of 10 ms. Note that memory use is computed for 64-bit events, with 32 bits for the timestamp. The clip is a "home" gesture (a hello-waving hand) from the NavGesture-walk database with a duration of around 1300 ms. It can be observed that image-like representations lead to memory footprint up one order of magnitude in this example.

4.6 COMPUTATIONAL COSTS

Operations that can be easily decomposed into a fixed number of equally sized components requiring the same type of computation are in principle portable to parallel processing [212]. However, algorithms where the workload is highly irregular, sparse and data-dependent, are substantially more challenging to port on clocked parallel processing environments, such as GPUs.

Because of the wide availability of GPUs and deep neural network programming languages, a large portion of existing work that generates frames from events (as introduced in sections 4.3 and 4.4) rely on the combined use of machine learning algorithms based on conventional deep neural networks and graphical processing units (GPUs). This comes at significant energy and computation costs that increase the gap between the mW energy and power efficiency of bioinspired neuromorphic hardware when compared to the

power consumption of available graphics cards with top performing models requiring anywhere between 110 and 270 Watts.

Comparing the use of event-based generated frames against pure incremental event-based computation can therefore only end up increasing the divide between the two approaches that is introduced at the memory level. In [192] authors exploited the sparsity on frames and inter-layer activation maps for CNNs execution for embedded systems by avoiding the multiply-accumulate (MAC) operation for null values, thus calling the method NullHop. The proposed hardware accelerator is used with normalized frames generated from an event-based camera by accumulating events at all spatial locations for a fixed number of events. Hence, the obtained frame has non-zero pixels only at locations where events were emitted. Nevertheless, because of frame-based computation the bottleneck of the system is the memory interface, making the performance of the method dependent on the memory hardware used. NullHop was tested with the VGG16 on ImageNet with a 67.5% top-1 accuracy using quantized weights and activations to 16 bits. It was also tested with a relative small CNN trained to classify hand symbols for the RoShamBo game, beating human opponents by recognizing the player's symbol with over 99% accuracy in less than 10 ms and a peak performance of 203 Gop/s/W using LPDDR3 memory. Perhaps a fair comparison of this method would be to consider a recent work from IBM research that introduced a fully neuromorphic pipeline made of an IBM TrueNorth chip and a DVS [13] to perform real-time gesture recognition. The system has been evaluated on the DvsGesture 10-class dataset, which was recorded using the DVS. However, even though the hardware is fully neuromorphic, the processing is frame-based, as they introduced a stochastic events-to-frame conversion to feed a CNN. This CNN performs around 1 billion convolutions per second (1 million convolution per tick, 1000 ticks per second). This results in one classification per tick, which are then averaged using a majority vote with a sliding window, for a final classification score of 96.49%. On the other hand, the 2-layer event-based architecture we presented in the previous Chapter performs around 250 millions convolutions for the whole clip (clips have a duration of 6-8 seconds), while achieving similar results in accuracy at 96.59% over the 10-class dataset with a single classification at the end of the clip. Moreover, IBM's stochastic frames are generated using a cascade of six temporal filters delaying events. The resulting output frame is the concatenation of all six filters outputs, which is nothing more than a stochastic integration of events over a duration of 81 ms. These frames are generated every millisecond. This automatically introduces a delay that corresponds to the integration time needed to generate a frame, plus the need to store an image-like representation for each filter. On the other hand, if one processes events in an event-based manner, the delay can be minimal, and the memory needed to dynamically store events greatly reduced, as shown in section 4.5.

4.7 GENERIC TIME ADAPTIVE MEMORY ARCHITECTURE FOR EVENT BASED PROCESSING

An optimal architecture for event-based computation must address two main intertwined requirements:

1. the retrieval of relevant local information around incoming events must match the high temporal precision of event-based cameras and ensure that computation can be carried out at the native temporal step of event-based cameras (1µs);

2. sparse and adaptive memory allocation following the scene-driven properties of event-based cameras and the temporal requirements of the used incremental algorithms.

The ideal memory structure that addresses these requirements is the Content-Addressable Memory (CAM), also known as Associative Memory. This type of memory structure allows for entire high-speed memory searches in a single clock cycle. CAM has been used extensively for several applications [213] including neural networks [214], and are particularly adapted for event-based processing. Unlike Random Access Memory (RAM), associative memory is content based, meaning events stored as content addressable can be accessed by performing a query for the content itself, and the memory retrieves the addresses where that data can be found. This query is parallel in nature by construction, and therefore orders of magnitude faster than conventional RAM. They are however more expensive to build because of the necessity of internal comparators and registers that require larger power consumption and more space. However, in the case of event-based sensing, because the required memory footprint is significantly reduced and because fast memory access is an absolute requirement, this make CAM particularly adapted to event-based processing.

We will introduce in this section a generic architecture for event-based processing in the context of machine learning with a classification task using temporal networks. These networks process events incrementally as they are output by the sensor. We will first introduce the concept of temporal networks, and in a second stage we will present the generic architecture followed by an implementation study of its hardware costs on FPGA.

4.7.1 Temporal Machine learning using Time-surfaces

Time-surfaces were first introduced in [196] and are local descriptors of the temporal activity in the spatial neighbourhood of an event. They allow a compact representation of both spatial and temporal information. They have been used in a variety of tasks and have been recently revisited and studied in several works such as [29, 215, 123]. We use the model presented in Chapter 3 that extends [196] to operate on low power processors in the context of mobile phones. We use this temporal convolution network relying on time surfaces to study how event-based algorithms could benefit from a dynamic, adaptive and sparse memory structure.

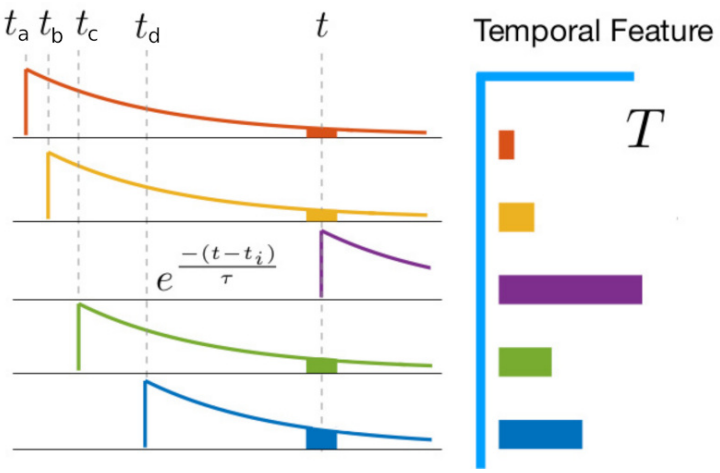


Figure 33: Principle of Temporal Context Representation. Five lines of information conveyed temporal events at different time t_a, t_b, t_c and t_d. A time-vector T is computed for each new incoming event (here the last event is in purple at time t) as a vector expressing temporal delays between events as normalized values. In this example, we use an exponential decay.

The general principle of a time-surface is shown in Fig.33 in the context of 1D input (thus generating a time-vector instead of a time-surface). Five temporal events are shown, each appearing at a particular time. The notion of an event here is generic, to simply introduce the basic concept behind time-surfaces. An event signals the presence of a particular activity at a precise location in time. The principle behind the method is to convert the relative timing between events occurring at different lines of information into normalized features that are invariant to the actual timing and emphasize only the temporal interval between past events and the last event that happens at the current time t, shown in purple in Fig.33. This method is event driven in the sense that if no event happens, nothing is computed. However, if an event occurs on a line of information at the current time, its time becomes the reference time from which we compute a temporal *context*, namely, how far in the past something happened on the other lines.

It is possible to convert the delay between the current event time t and other events shown with different colors and appearing at times t_a, t_b, t_c and t_d into normalized values. Here we use an exponentially decaying function with a time-constant τ, in order to map delays between 0 and 1, such that the timing of the events that are closer to the reference spike give values close to 1 while those occurring earlier tend closer to 0, as shown in Fig. 33. Thus, for each incoming event, we can define a time-vector T of dimension n, where n is the current number of temporal information lines (5 for the example shown in the Figure), computed as : $_j = exp(\frac{-|t-t_j|}{\tau})$. Another kernel could be used to map timestamps into a chosen interval. In the following, we use the linear decay kernel used in the previous chapter. We first present the general formulation for time-surfaces when working directly with the stream of event, and then how we can take advantage of the newly introduced memory to ease their computation, as some of the processing is already done at the memory level.

General Computation of Time-surfaces

The time-surface related to the event e_k, can be computed from the stream of events $e_0, ..., e_k$ received from the event-based camera. First we define the Time-context $\mathcal{T}_k(\mathbf{u}, p)$ of the event e_k as the difference between e_k time of arrival and the timestamps of its most recent neighbours. The square neighbourhood Σ_k has a dimension of (2R + 1) x (2R + 1). It is centered on e_k, of spatial coordinate $\mathbf{u}_k = [x_k, y_k]$. Mathematically the Time-context can be expressed as:

$$\mathcal{T}_k(\mathbf{u}, p) = \{t_k - t | t = max\{t_j | \mathbf{u}_j \in \Sigma_k, p_j = p\}\} \tag{30}$$

The Time-surface $S_k(\mathbf{u}, p)$ associated to the event e_k, is obtained by applying a linear decay kernel of time-constant τ to the Time-context $\mathcal{T}_k$:

$$S_k(\mathbf{u}, p) = \begin{cases} 1 - \frac{\mathcal{T}_i(u,p))}{\tau}, & \text{if } \mathcal{T}_i(u, p) < \tau \\ 0, & \text{otherwise} \end{cases} \tag{31}$$

This construction makes time-surfaces a compact description of the spatio-temporal neighborhood of an event e_k. Canonical time-surfaces can be learned and used as features in a pattern recognition task.

Learning Time-surfaces

The Hierarchy of Time Surface is an event-driven algorithm that operates on each incoming event introduced in [196] and extended in Chapter 3. The general idea is shown in Fig. 34. For each incoming event, it is possible to compute a time-surface that is matched against a bank of learnt prototypes. A new *pattern* event with the same **x** address and timestamp t is generated, but its polarity p is updated to become the ID of the matched prototype. This *pattern* event encodes a pattern instead of the ON/OFF polarity of *visual* events. Pattern events at the output of a layer can be fed to the subsequent layer to be processed in a similar way. A higher layer however combines patterns from the previous layer, leading to more sophisticated patterns being encoded. The pattern of output events can be used as a feature for classification.

The first layer will encode only basic spatio-temporal patterns such as optical-flow and shape of the edge. Adding a second layer will increase the pattern complexity as it will be able to create new features as spatio-temporal combinations of patterns from the first layer. Subsequent layers operating at increasingly higher timescales allow for more and more sophisticated patterns.

4.7.2 A Generic Memory Architecture

Fig.35 introduces the general architecture for implementing an event based deep network using time-surfaces. The general idea is to parallelize and speed up computation specifically when fetching previous events in the neighborhood of the incoming event.

An event-based camera (A) outputs an event (B) stored at the top the list of an associative memory containing past events (C). The associative

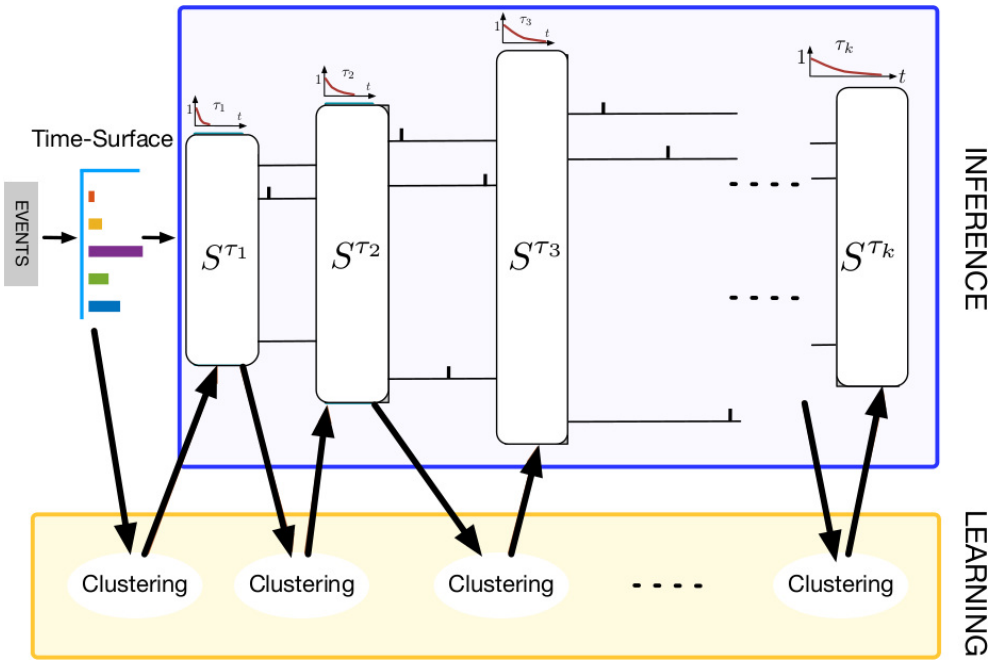


Figure 34: (A) The principle of deep temporal networks: an incoming event triggers the computation of a temporal feature, that will be compared to existing learnt features S^{τ_1}. The closest learnt feature will emit an event that will trigger the computation of a new temporal feature on a larger integration time τ_2. New events generated at layer one can be used to compute a new feature using a larger integration time τ_2. The same process is performed at layer 2 that will generate new events that can be sent to a next layer with a larger integration time (τ_3). The system can be scaled up for an increasing number of layers with increasing integration times. Learning is carried out at each layer using online clustering to extract the most representative features.

memory is conventionally based on three modules: the memory controller, the memory core and the streamer. The memory controller is in charge of two functions, it receives the input event and stores it in the memory core, organizing the memory content according to the timestamps. This controller uses the timestamps to remove events from memory that are too old and not relevant anymore. The second module is the memory core, where events data is stored. This memory is implemented using a bank of cells, so data can be accessed in parallel in an associative way, reducing the latency. Most of digital architectures use CAM which are accessed cycle-by-cycle. Finally, the streamer organizes the data in a square neighbourhood (or row) to be processed as shown in (D-E).

The proposed architecture requires 1-2 clock cycles to stream (or output) the data. Since the memory controller sends neighbours' data to the streamer at the same time as it reads from the memory core in one clock cycle (thanks to the parallel comparator-architecture of CAM), the streamer module needs another clock cycle to organize the data format. The neighborhood events' around the incoming event are shown in cyan. In this work, memory cells have been implemented as registers that can be read/write in 1 clock cycle. It then becomes possible to compute the time-surface from the temporal

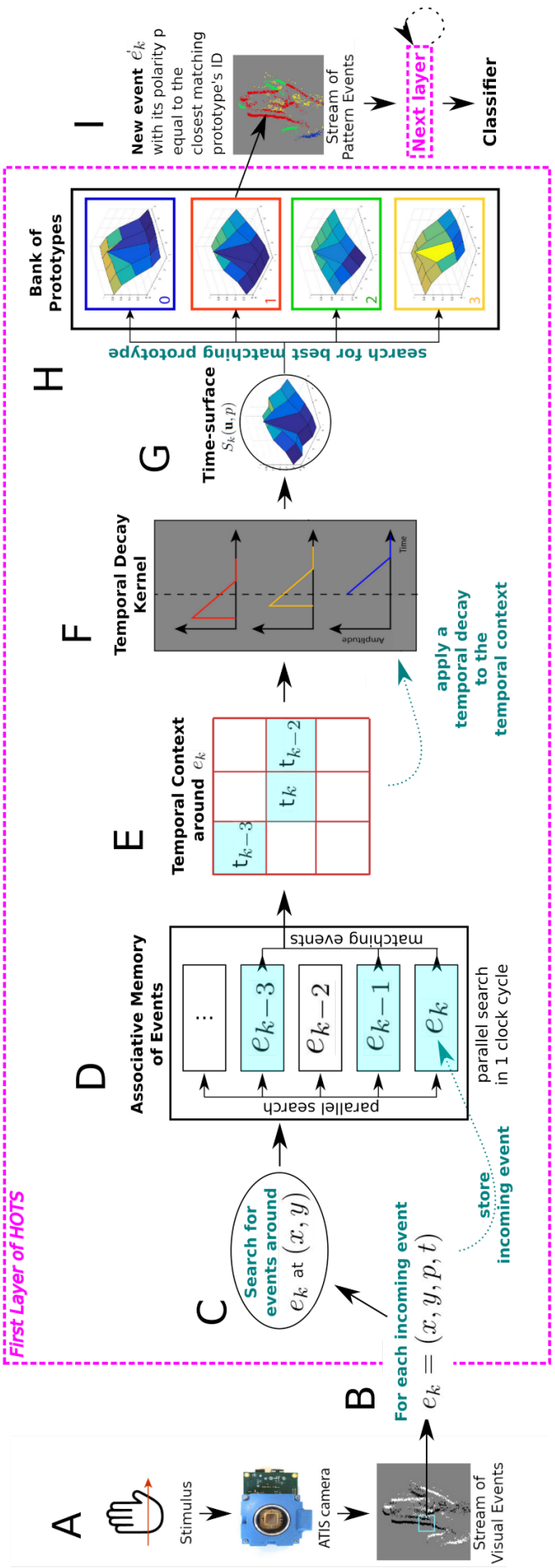


Figure 35: From the visual stimulus to the classification using a layer of HOTS as a feature extractor. (A) The sensor reacts to the stimulus, and emits a stream of visual events. (B) The incoming event e_k is stored into the associative memory. (C-D-E) Neighboring events are retrieved from the memory in order to obtain the temporal context. (F-G) The time-surface is computed from this context by normalizing delays between events using a temporal kernel. (H) The time-surface is matched against a bank of known patterns, called prototypes. (I) The polarity of the event is updated to represent the matched prototype. The event can be either passed on to the next layer working at another spatio-temporal scale or can be used as a feature by a classifier.

context, by applying a decaying kernel (linear or non linear) shown in (E-F-G).

The computed time-surfaces needs then to be compared with the stored prototypes time-surfaces. The prototypes could also be stored in a CAM as shown in (H). It would allow to fetch the most representative ones, thus generating a new event tagged by the polarity of the matched prototype time-surfaces as shown in (I). The same process can be scaled up to higher layers. The architecture and the use of associative memory solves greedy computation issues that usually require incremental high complexity algorithms. Using CAM is highly beneficial from a latency perspective because of its parallelization improvement on data searching. Nevertheless, these memories cannot be oversized despite the increment on resources for a viable implementation. Like cache memories for processors, CAM sizes must be adjusted for the best compromise.

4.7.3 "In-memory" partial computation of Time-surfaces

The new dynamic memory proposed in this Chapter offers the advantage of handling some of the needed computation in order to obtain a time-surface. If this memory is used, it eases the time-surface computation process as some operations are already performed "in-memory". The following equations represent particular cases of the above formulation of time-surfaces presented in Eq. 30 and Eq. 33.

Once again, we define the spatial neighborhood Σ_k as a square of dimension (2R + 1) x (2R + 1), centered on $\mathbf{u}_k$. Because the dynamic memory keeps only the most recent event at each spatial location, the *max* operation in Eq. 30 can be carried out at the memory level. This allows for the simplification of Eq.30to :

$$\mathcal{T}_k(e_j) = \{t_k - t_j | \mathbf{u}_j \in \Sigma_k, p_j = p_k\} \tag{32}$$

Furthermore, because the dynamic memory uses a time-window T, by matching τ to T, the time-surface $S_k(e_i)$ associated with the event e_k, is obtained by directly applying the decay kernel of time-constant τ to the Time-context $\mathcal{T}_k$ as only the events younger than $(t_k - \tau)$ are available in the memory:

$$S_k(\mathbf{u}, p) = 1 - \frac{\mathcal{T}_k(\mathbf{u}, p))}{\tau} \tag{33}$$

This results in some operations needed to compute the time-surface to be processed "in-memory" instead of using additional processor resources.

4.7.4 Hardware Implementation Study

We implement the proposed memory model to be used with the event-based algorithm from Chapter 3 to recognize gestures from the NavGestures datasets. The aim behind these experiments is to provide an overview of the viability of the proposed memory model in terms of latency and memory usage. We will consider an FPGA implementation, the memory requirement

per events being 64-bit, where 32-bit stores the timestamp and 32-bit stores the address (x,y) and the pattern.

We performed two experiments, the first one using a single layer on a FPGA, and second using two layers. These experiments have been performed in simulation on an FPGA after the implementation phase.

For the singe layer implementation, the memory model parameters are:

- τ = 10ms (integration time)
- R = 2 (radius)
- N = 8 (number of prototypes)

The classification output using the dynamic memory was compared to the results obtained using the static memory. It must be noted that a memory that stores all pixel states should give more precision than storing only most recent events. The accuracy error was measured for different values of the time-window T and the memory size cap.

Large values of T implies that more events will be stored in memory, thus increasing the memory size. On the other hand, smaller T values store only the most recent events, meaning less information, leading to a worse classification accuracy. Fig.36 shows how the memory size and the accuracy error evolves when the time-window T increases. As an example, for the NavGesture dataset used in this experiment, a T value of 15 ms with a memory of 756 Kb gives an error of less than 2% compared with the original implementation that uses 2.280 Kb.

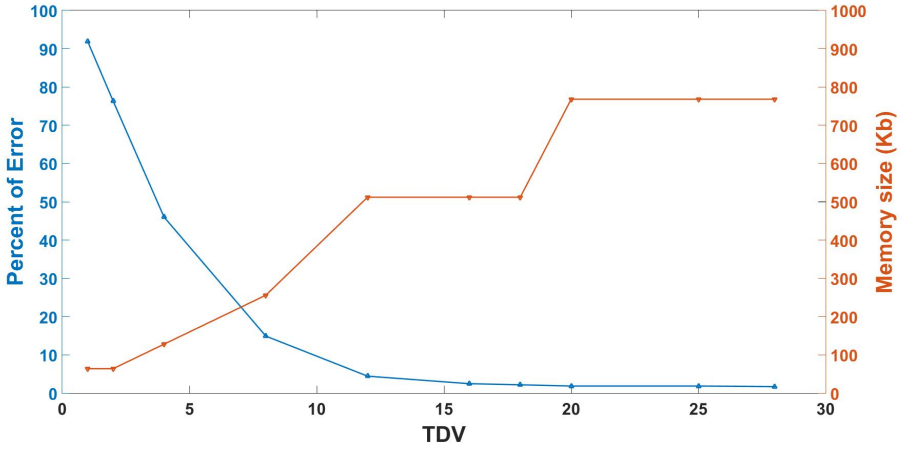


Figure 36: NavGesture dataset: Percent of Error (blue) and Memory Size (orange) as function of the time-window T

The DAVIS 240C event-based sensor [216] requires 16 bits to encode the spatial location and 1 bit for the polarity. On the other hand, the ATIS requires 18 bits (9 for x address, 8 for y address and 1 for the polarity bit). We have considered that both sensors use a resolution of 32 bits for timestamps.

Taking into consideration that the worst case is a 756 Kb memory, for a single layer implementation the memory model of this work manage to reduce the memory space by 77% and 44% for ATIS and DAVIS 240C sensors respectively.

Compared to the previous experiment, a 2-layer network implementation requires larger memory cells, as apart from the input memory for the sensor, it also needs to store the output pattern events of the first layer which are fed to the second layer. The purpose is to determine the minimum required dynamic memory in order to obtain the same surfaces as with standard static memory, therefore, there is no accuracy loss. Table 8 shows a summary of the 2-layers networks used in this experiment.

Network	1			Layer 2		
	N_1	R_1	τ_1	N_2	R_2	τ_2
A	8	2	10ms	8	2	100ms
B	8	2	10ms	8	2	200ms
C	8	2	10ms	8	2	400ms
D	8	2	10ms	8	2	800ms

Table 8: Characteristics of the 2-layers networks used in this experiment.

Larger values of T implies longer integration time, and thus larger memories are needed to avoid the loss of significant events. In the 1-layer network, only one memory is needed to store the timestamps of the input events coming from the sensor. On the other hand, in 2-layers networks the pattern events between each layer must also be stored in memories, grouping the events by patterns (polarity). Therefore, as well as the required memory for input events, it also needs as many supplementary memories as there are prototypes on the first layer. The T value of these memories are set to match the τ_2 value of the second layer.

The size of each memory used in these networks to avoid losing accuracy was set to 1024Kb, assuming 32 bits for events addresses and 32 bits for timestamps. Since in **A** and **B** layers the memory is never full. In layers **C** and **D**, due the τ_2 value is higher (400ms and 800ms). These values have been chosen to set an upper boundary and show how the architecture would deal with extreme cases of deep layers where the integration time can be large. The total memory of 9,216Kb (one 1,024kb memory for the first layer, and eight 1,024 memories for the second layer) is sufficient to run this network without losing precision. Fig. 37 shows the memory needed to implement this network using dynamic memory for the ATIS and the Davis 240C sensor compared with an implementation using static memory.

Regarding latency, one can plot a roof-line diagram, which corresponds to the number of operations versus the memory access. First it is important to highlight the number of operation performed by the feature extraction algorithm. Following Eq 32 and 31 a division and two subtraction operations are needed to compute the time-surface. In addition, the Euclidean distance requires one addition, two subtractions, two multiplications and the square root per cycle and prototype. Eq. 34 gives the total number of operations needed for one layer.

$$Op = (\overbrace{(2R+1*2R+1)}^{\text{Square neighbourhood}} *(\overbrace{3}^{\text{TS}}) + (\overbrace{6}^{\text{ED}} *N)) \tag{34}$$

Figure 38 shows the roofline diagram for the whole processing. The peak of performance is reached when all neighbours have been read from the

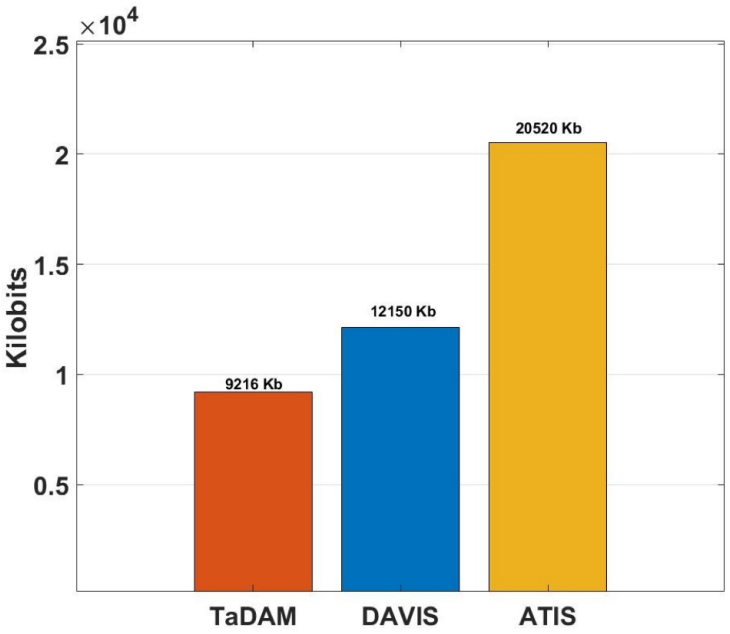


Figure 37: Memory consumption of event-based memory with memory consumed by RAM memory for DAVIS 240c and ATIS sensors.

memory, generating the time surfaces and compared them with the prototypes of both layers. The slope of the original static-memory implementation depends on the number of access to obtain all the neighboring events. In this original implementation, memory is read sequentially. Therefore, latency to start the computation increase with the radius R of the patch. On the other hand, the memory model presented in this work requires only two clock cycles. Since data is ready for computation in two clock cycles, it can reach its maximum performance fast.

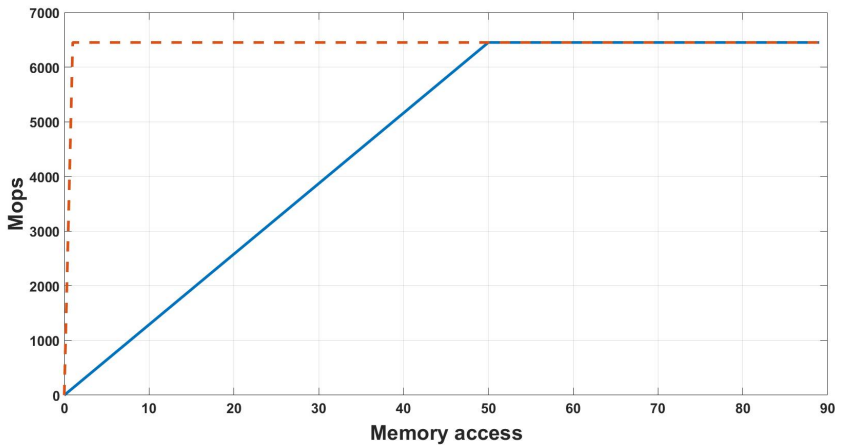


Figure 38: Roofline of HOTS algorithms using both memory models.

4.8 DISCUSSION AND CONCLUSION

This chapter focused on studying the limitations of the current trend of converting events to frames. Although critical of this approach, this is however an expected stage for this field. Although neuromorphic engineering is growing exponentially since the last couple of years with event-based cameras commercially available to laymen, there is currently no off-the-shelves hardware able to process events incrementally at their native temporal resolution. Beyond the difficulty of diving into the novelty of temporal processing and breaking free from frames, and although several work have shown that incremental algorithms are superior in numerous ways to those using frames from events even using conventional processors, the current availability, low cost and ease of use of hardware such as GPU tend to emphasize even more the easy and narrow path of recycling techniques from conventional frame-based computer vision.

Perhaps the solution is to derive a new generation of hardware such as the associative memory introduced in this work. The event-based dynamic architecture based on associative memory allows to reduce memory load and computation time for event-based algorithms. It is adapted to make full use of their native sparseness properties while allowing incremental computation with no unnecessary memory payload. The use of in-memory computation allows for an optimized memory access while addressing heavy computation required for pattern matching and neighborhood fetching around incoming events.

This architecture can be applied to any event based algorithm. The specific application to machine learning using deep temporal networks has been chosen as a study case as it represents the most unfavourable scheme to generating frames from events because of the heavy requirement of conventional machine learning from a resource perspective. The presented architecture used two associative memory, the second being specific to pattern matching for deep temporal networks using time surfaces. It could however be replaced for more general event-based incremental computations by a low power conventional processor to perform conventional computations around incoming events. There are several available incremental event-based algorithms such as optical flow, tracking, etc that are already operating efficiently on conventional CPU that would greatly benefit from this approach where accessing past event and fetching spatio-temporal contexts is their major limitation.

It is expected that similar architectures will become essential in the near future as the spatial resolution of event-based sensors is increasing rapidly as more traditional industrial players are becoming involved in their development and production [4]. The technique of generating frames and recycling frame-based algorithms as shown by this work will result in even more dramatic waste of resources for higher spatial resolution and/or and lower latency sensors. Recent sensors such as the one developed by Samsung [4] have a huge throughput compared to older neuromorphic sensors such as the academically developed prototypes - DVS or ATIS - that can reach 300 MEvents/Second.

As shown by the hardware FPGA study, the architecture would benefit from a parallel memory access. One possible solution is to implement each position of the memory as a register, which can be accessed in parallel, and integrate a bank of comparators to efficiently select the desired content(s). However, registers require more power and physical area to implement than RAM cells. An alternative solution could be found in the emerging memristor technology, which are currently demonstrated to be a good approach to parallel processing systems [217]. This property makes memristors ideal for neuromorphic systems spikes grid where neurons have to communicate with their neighbours in parallel [218]. Memristors technology would allow to implement the presented memory model using fewer resources, reducing the memory bottlenecks, thus improving the latency and boosting neuromorphic incremental computation.

5 DISCUSSION & CONCLUSION

The objective of this thesis was to develop efficient event-based methods using the precise timing of events in feature extraction and pattern recognition. We started from extracting low-level event-based features and went up to building a full pattern recognition pipeline, and we also pushed further the event-based computation paradigm, by providing a dedicated hardware proposal in the form of a "dynamic" associative memory architecture. This memory offers both a more efficient storage and faster retrieval of events but moreover, allows for "in-memory" computation. This thesis benefited from the ECOMODE project, that provided an interesting test-bed for efficient event-based acquisition and processing, as it aimed for a gesture recognition pipeline embarked on a standard smartphone. This resulted in a pioneering step in mobile vision-based dynamic gesture recognition. All methods and concepts presented in this thesis are event-driven, being either semi-accumulative or non-accumulative. They constitute a demonstration that it is possible to develop efficient computer vision and machine learning techniques that are event-based and do not rely on the typical frame-based paradigm when considering dynamic data. The ECOMODE smartphone is a blatant example: along with the gesture recognition, the system allows dynamic background suppression. This could not be carried out using a conventional frame-based approach – at least not with the available resources – even with OpenCV libraries optimized for Android. This thesis must not be mistaken as a pamphlet against frame-based processing, that has achieved significant milestones, even surpassing human performance in some tasks. But frame-based techniques are suited for static data. As for machine learning, we showed that when using features that truly encode the dynamics, it is possible to outperform very sophisticated architectures that rely on unsuited "static" features and require an extravagant number of examples to be trained. Also, we demonstrated with the Motion-based Feature that it was possible to detect static properties, in our case corners, using dynamical information.

Event-based sensing and processing is antagonistic to the frame-based approach in several ways, and the idea to distribute computing task along the different components of the system is primordial. The proposed memory allows to perform computation on the stream of events. In a first step, this memory could be added to standard PCs as an improvement for running event-based algorithms. But then, if the philosophy behind event-based processing is that the computation should be distributed across the system, why should such systems be implemented on a *von Neumann* architecture, which is not known for being very efficient at massively parallel processing? A fully distributed approach, where each unit of the system performs an incremental part of the total computation, is a promising path for overcoming memory and processing bottlenecks. This would require a highly

parallel computational architecture, able to preserve the precise timing of events as they flow through small processing units. Our laboratory developed a parallel and non-*von Neumman* computational architecture: the Spike Time Interval Computational Kernel (STICK) [111], that encodes values as the time interval between two events. It is a radical proposal for neuromorphic computation as it totally intertwines memory and computation in this fully time-based approach, making no distinct partitioning between the two. Since this architecture makes full use of the precise timing of events, while operating in an asynchronous manner, making it a possible candidate for a yet-to-build general purpose event-based computer.

The design of non-accumulative methods and semi-accumulative algorithms, is at the core of numerous recent event-based methods, including those presented in this work. But as for now, most event-based algorithms do run on standard PCs, and events are received from the camera through USB-like buses. In order to prove the superiority of the event-based paradigm in tasks involving dynamic data, and in the absence of widely accessible neuromorphic hardware, event-based algorithms must be designed to run as efficiently as possible on standard computers. In this context, events are received and processed in a sequential and ordered manner and this is the reason why semi-accumulative methods are prone to partial- and over-processing. More explicitly, as semi-accumulative methods are triggered by each new incoming event, they will process a local neighborhood, and they will be triggered again when another event happens in this neighborhood. The local neighborhood will hence be processed again, with only one more event. This asks if that neighborhood should have been processed at the first event: if not, it resulted in partial-processing as the available information was still "incomplete". The same can be asked at the second event, and if the answer is no, then over-processing occurred, resulting in redundant information. This is why some algorithmic aspects of semi-accumulative methods have to be further investigated. In fact, this current issue in semi-accumulative methods can be seen as a sampling problem: the algorithm "samples" the stream of events, which asks the question whether each new event should trigger the full processing pipeline? Such asynchronous sampling lacks a theoretical background, and is still looking for its equivalents of Shannon and Nyquist theorems. There is hence a need for a framework that would, in a first step, allow to systematically assess partial- and over-processing that can occur in semi-accumulative methods, and would be an interesting basis to improve these techniques.

Time-surfaces and HOTS-like hierarchical models have been an important part of this Ph.D. work. In these models, higher layers in the hierarchy encode more and more sophisticated patterns. This means that one event can represent multiple events from the previous layer. Partial- and over-processing are currently major limitations. Because of over-processing, the model outputs several similar events to encode the same stimulus. This translate in redundant information at the input of the next layer. Because of partial-processing, the model outputs several events that encode "incomplete" information (which is also an issue during learning, as it makes the model learn "incomplete" patterns). These two issues prevent layers to output a sparser stream of events, preventing a more and more compressed

representation of the visual scene in higher-level layers. We proposed a first heuristic based on the "completeness" of time-surfaces. A coarser approach could be the deletion of older events that have already been processed and used in a time-surface. Further solutions could include an inhibitory mechanism that would prevent the immediate extraction of a second time-surface.

Maybe some of these issues will be inherently solved by future highly parallel event-based hardware, as events that happen will be all transmitted and processed in parallel, instead of sequentially as it is currently the case. All events' information would be merged at some higher level in the system, and it would dismiss the need for the construction of an artificial neighborhood at each new event like in semi-accumulative methods. The next challenge for event-based processing will be to fundamentally transform these algorithms, which are adapted for sequential processing, to dedicated fully parallel neuromorphic hardware, making sure to remove the bricks that belong to the old paradigm. As more and more neuromorphic chips are being developed by big industrial players, such as IBM's TrueNorth or Intel's Loihi, dedicated neuromorphic hardware could become widely available in a foreseeable future. This also sends a clear signal that neuromorphic computing is a promising path.

Lastly, the ECOMODE project punctuated this Ph.D. work, and I am pleased that we could deliver a working smartphone prototype with its gesture recognition module. This prototype still need improvement, notably in areas such as the gesture segmentation. This is currently done by using an activity threshold, but by adding information received from the Inertial Measurement Unit (IMU) of the smartphone, the robustness could be greatly increased, especially against jerky movements. However, the prototype was well received by its targeted end-users, and more specifically the elderly.

BIBLIOGRAPHY

[1] Thomas P Hughes. "Technological momentum". In: *The social construction of technological systems: New directions in the sociology and history of technology* (1987).

[2] P Lichtsteiner, C Posch, and T Delbrück. "A 128 x 128 120 dB 15 us Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE Journal of Solid-State Circuits* 43.2 (Feb. 2008), pp. 566–576. ISSN: 0018-9200. DOI: 10.1109/JSSC.2007.914337.

[3] 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. ISSN: 0018-9200. DOI: 10.1109/JSSC.2010.2085952.

[4] B. Son et al. "A 640x480 dynamic vision sensor with a 9µm pixel and 300Meps address-event representation". In: *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. Feb. 2017, pp. 66–67. DOI: 10.1109/ISSCC.2017.7870263.

[5] Alexandre Marcireau et al. "Event-Based Color Segmentation With a High Dynamic Range Sensor". In: *Frontiers in Neuroscience* 12 (Feb. 2018). DOI: 10.3389/fnins.2018.00135.

[6] Guang Chen et al. "FLGR: Fixed length GISTS representation learning for RNN-HMM hybrid-based neuromorphic continuous gesture recognition". In: *Frontiers in neuroscience* 13 (2019), p. 73.

[7] Xavier Lagorce et al. "Hots: a hierarchy of event-based time-surfaces for pattern recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 39.7 (2016), pp. 1346–1359.

[8] R. Benosman et al. "Event-Based Visual Flow". In: *IEEE Transactions on Neural Networks and Learning Systems* 25.2 (Feb. 2014), pp. 407–417. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2013.2273537.

[9] Yijing Watkins et al. "Sparse coding enables the reconstruction of high-fidelity images and video from retinal spike trains". In: *Proceedings of the International Conference on Neuromorphic Systems*. 2018, pp. 1–5.

[10] Lin Wang, Yo-Sung Ho, Kuk-Jin Yoon, et al. "Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10081–10090.

[11] Cedric Scheerlinck et al. "Fast image reconstruction with an event camera". In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 156–163.

[12] Timo Stoffregen et al. "How to Train Your Event Camera Neural Network". In: *arXiv preprint arXiv:2003.09078* (2020).

[13] Arnon Amir et al. "A low power, fully event-based gesture recognition system". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7243–7252.

[14] Rohan Ghosh et al. "Spatiotemporal Filtering for Event-Based Action Recognition". In: *arXiv preprint arXiv:1903.07067* (2019).

[15] Alex Zihao Zhu et al. "EV-FlowNet: Self-supervised optical flow estimation for event-based cameras". In: *arXiv preprint arXiv:1802.06898* (2018).

[16] Chengxi Ye et al. "Unsupervised learning of dense optical flow, depth and egomotion from sparse event data". In: *arXiv preprint arXiv:1809.08625* (2018).

[17] R. Ghosh et al. "Real-time object recognition and orientation estimation using an event-based camera and CNN". In: *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*. 2014, pp. 544–547.

[18] Aaron Chadha et al. "Neuromorphic Vision Sensing for CNN-based Action Recognition". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 7968–7972.

[19] Diederik Paul Moeys et al. "Steering a predator robot using a mixed frame/event-driven convolutional neural network". In: *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. IEEE. 2016, pp. 1–8.

[20] Joubert Damien, Konik Hubert, and Chausse Frederic. "Convolutional Neural Network for Detection and Classification with Event-based Data". In: (2019).

[21] R Baldwin et al. "Event Probability Mask (EPM) and Event Denoising Convolutional Neural Network (EDnCNN) for Neuromorphic Cameras". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1701–1710.

[22] Thomas S Kuhn. *The structure of scientific revolutions*. University of Chicago press, 1962.

[23] Christoph Posch, Daniel Matolin, and Rainer 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.

[24] David Reverter Valeiras et al. "An asynchronous neuromorphic event-driven visual part-based shape tracking". In: *IEEE transactions on neural networks and learning systems* 26.12 (2015), pp. 3045–3059.

[25] David Reverter Valeiras et al. "Event-Based Line Fitting and Segment Detection Using a Neuromorphic Visual Sensor". In: *IEEE transactions on neural networks and learning systems* 30.4 (2018), pp. 1218–1230.

[26] Xavier Lagorce et al. "Asynchronous event-based multikernel algorithm for high-speed visual features tracking". In: *IEEE transactions on neural networks and learning systems* 26.8 (2014), pp. 1710–1720.

[27] Elias Mueggler, Basil Huber, and Davide Scaramuzza. "Event-based, 6-DOF pose tracking for high-speed maneuvers". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 2761–2768.

[28] Xavier Clady et al. "A Motion-Based Feature for Event-Based Pattern Recognition". In: *Frontiers in neuroscience* 10 (2017), p. 594.

[29] Jacques Manderscheid et al. "Speed invariant time surface for learning to detect corner points with event-based cameras". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10245–10254.

[30] Bharath Ramesh et al. "Dart: distribution aware retinal transform for event-based cameras". In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[31] Saeed Afshar et al. "Investigation of event-based surfaces for high-speed detection, unsupervised feature extraction, and object recognition". In: *Frontiers in neuroscience* 12 (2019), p. 1047.

[32] K. Mikolajczyk and C. Schmid. "A performance evaluation of local descriptors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.10 (2005), 1615–Đ1630.

[33] I. Laptev. "On Space-Time Interest Points". In: *International Journal of Computer Vision* 64.2/3 (2005), pp. 107–123.

[34] F. Mokhtarian and F. Mohanna. "Performance evaluation of corner detectors using consistency and accuracy measureness". In: *Comput. Vis. Image Understand.* 102.1 (2006), 81–Đ94.

[35] P. Moreels and P. Perona. "Evaluation of features detectors and descriptors based on 3D objects". In: *International Journal of Computer Vision* 73.3 (2007), pp. 263–284.

[36] Arturo Gil et al. "A comparative evaluation of interest point detectors and local descriptors for visual SLAM". In: *Machine Vision and Applications* 21.6 (2010), pp. 905–920.

[37] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. "Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking". In: *International Journal of Computer Vision* 94 (2011), pp. 335–360.

[38] Timo Dickscheid, Falko Schindler, and Wolfgang Förstner. "Coding Images with Local Features". In: *International Journal of Computer Vision* 94.2 (2011), pp. 154–174.

[39] Tobi Delbrück et al. "Activity-driven, event-based vision sensors". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE. 2010, pp. 2426–2429.

[40] Christoph Posch. "Bioinspired vision sensing". In: *Biologically Inspired Computer Vision*. Wiley Online Library, 2015, pp. 11–28.

[41] 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: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*. May 2010, pp. 2430–2433. DOI: 10.1109/ISCAS.2010.5537150.

[42] P. Lichtsteiner, C. Posch, and T. Delbruck. "A 128*128 120dB 15us latency asynchronous temporal contrast vision sensor". In: *IEEE Journal of Solid State Circuits* 43.2 (2008), pp. 566–576.

[43] T. Serrano-Gotarredona and B. Linares-Barranco. "A 128x128 1.5% 20 Contrast Sensitivity 0.9% 20 FPN 3 μs Latency 4 mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Preamplifiers". In: *IEEE Journal of Solid-State Circuits* 48.3 (Mar. 2013), pp. 827–838. ISSN: 0018-9200. DOI: 10.1109/JSSC.2012.2230553.

[44] Thibaud Debaecker, Ryad Benosman, and Sio H Ieng. "Image Sensor Model Using Geometric Algebra: From Calibration to Motion Estimation". In: *Geometric Algebra Computing* (2010), pp. 277–297.

[45] Andrea Censi et al. "Low-latency localization by Active LED Markers tracking using a Dynamic Vision Sensor". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 891–898.

[46] Tobi Delbruck and Manuel Lang. "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor". In: *Frontiers in neuroscience* 7 (2013), p. 223.

[47] Xavier Lagorce, Sio-Hoi Ieng, and Ryad Benosman. "Event-based features for robotic vision". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 4214–4219.

[48] Xavier Clady et al. "Asynchronous visual event-based time-to-contact". In: *Frontiers in neuroscience* 8 (2014).

[49] Zhenjiang Ni et al. *Haptic Feedback Teleoperation of Optical Tweezers*. John Wiley and Sons, 2014.

[50] Moritz Milde et al. "Bioinspired event-driven collision avoidance algorithm based on optic flow". In: *Event-based Control, Communication, and Signal Processing (EBCCSP)*. Krakow, June 2015. DOI: 10.1109/EBCCSP.2015.7300673.

[51] David Drazen et al. "Toward real-time particle tracking using an event-based dynamic vision sensor". In: *Experiments in Fluids* 51.5 (2011), pp. 1465–1469.

[52] Zhenjiang Ni et al. "Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras". In: *Neural Computation* 20.4 (2015), pp. 1–29.

[53] Paul Rogister et al. "Asynchronous Event-Based Binocular Stereo Matching". In: *Neural Networks and Learning Systems, IEEE Transactions on* 23.2 (2012), pp. 347–353.

[54] Joao Carneiro et al. "Event-based 3D reconstruction from neuromorphic retinas". In: *Neural Networks* 45 (Sept. 2013), pp. 27–38.

[55] L. A. Camuas-Mesa et al. "On the use of orientation filters for 3D reconstruction in event-driven stereo vision". In: *Frontiers in neuroscience* 8 (2014).

[56] M. Firouzi and J. Conradt. "Asynchronous Event-based Cooperative Stereo Matching Using Neuromorphic Silicon Retinas". In: *Neural Processing Letters* (2015), pp. 1–16.

[57] Xavier Clady, Sio-Hoi Ieng, and Ryad Benosman. "Asynchronous event-based corner detection and matching". In: *Neural Networks* 66 (2015), pp. 91–106.

[58] J. Perez-Carrasco et al. "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing - application to feedforward convnets". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.11 (2013), pp. 2706–2719.

[59] Garrick Orchard et al. "HFirst: A Temporal Approach to Object Recognition". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 37.10 (2015).

[60] Himanshu Akolkar et al. "What can neuromorphic event-driven precise timing add to spike-based pattern recognition?" In: *Neural computation* 27.3 (2015), pp. 561–93.

[61] Garrick Orchard et al. "Real-time event-driven spiking neural network object recognition on the SpiNNaker platform". In: *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*. IEEE. 2015, pp. 2413–2416.

[62] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. "Training Deep Spiking Neural Networks using Backpropagation". In: *Frontiers in Neuroscience; Neuromorphic Engineering* (Nov. 2016).

[63] Jun Haeng Lee et al. "Real-time gesture interface based on event-driven processing from stereo silicon retinas". In: *Neural Networks and Learning Systems, IEEE Transactions on* 25.12 (2014), pp. 2250–2263.

[64] Xavier Lagorce et al. "Asynchronous event-based multikernel algorithm for high-speed visual features tracking". In: *Transactions on Neural Networks and Learning Systems* (2014).

[65] Elias Mueggler et al. "Towards evasive maneuvers with quadrotors using dynamic vision sensors". In: *Mobile Robots (ECMR), 2015 European Conference on*. IEEE. 2015, pp. 1–8.

[66] Ryad Benosman et al. "Event-based visual flow". In: *Neural Networks and Learning Systems, IEEE Transactions on* 25.2 (2014), pp. 407–417.

[67] Garrick Orchard and Ralph Etienne-Cummings. "Bioinspired Visual Motion Estimation". In: *Proceedings of the IEEE* 102.10 (2014), pp. 1520–1536.

[68] Tobias Brosch, Stephan Tschechne, and Heiko Neumann. "On event-based optical flow detection". In: *Frontiers in neuroscience* 9 (2015).

[69] Elias Mueggler et al. “Lifetime estimation of events from Dynamic Vision Sensors”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 4874–4881.

[70] David H Hubel and Torsten N Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *The Journal of physiology* 160.1 (1962), pp. 106–154.

[71] David H Hubel and Torsten N Wiesel. “Receptive fields and functional architecture of monkey striate cortex”. In: *The Journal of physiology* 195.1 (1968), pp. 215–243.

[72] NJ Priebe, SG Lisberger, and JA Movshon. “Tuning for spatiotemporal frequency and speed in directionally selective neurons of macaque striate cortex”. In: *J Neurosci* 26 (2006), pp. 2941–2950.

[73] G. A. Orban, J. de Wolf, and H. Maes. “Factors influencing velocity coding in the human visual system”. In: *Vision research* 24.1 (1984), pp. 33–39.

[74] S. Kime et al. “Exploring speed discrimination of visual stimuli at a high frame rate”. In: *Annual Meeting of the Society For Neuroscience(SFN)*. 2014.

[75] Sihem Kime et al. “Psychophysical assessment of perceptual performance with varying display frame rates”. In: *Journal of Display Technology* 12.11 (2016), pp. 1372–1382.

[76] MCW. van Rossum. “A novel spike distance”. In: *Neural Comput* 13 (2001), pp. 751–763.

[77] S. Schreiber et al. “A new correlation based measure of spike timing reliability”. In: *Neurocomputing* 52 (2003), pp. 925–931.

[78] Wyeth Bair and J. Anthony Movshon. “Adaptive Temporal Integration of Motion in Direction-Selective Neurons in Macaque Visual Cortex”. In: *The Journal of Neuroscience* 24.33 (2004), pp. 7305–7323.

[79] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[80] Seung Park et al. “Image Corner Detection Using Radon Transform”. In: *Computational Science and Its Applications*. Ed. by Antonio Lagano et al. Vol. 3046. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 948–955. ISBN: 978-3-540-22060-2.

[81] H. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. Tech. rep. CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University and doctoral dissertation, Stanford University, 1980.

[82] F. Mokhtarian and R. Suomela. “Robust image corner detection through curvature scale space”. In: *IEEE Trans on Pattern Analysis and Machine Intelligence* 20.12 (1998), pp. 1376–1381.

[83] C. Harris and M. Stephens. “A combined corner and edge detection”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* (1988), pp. 147–151.

[84] E. Rosten and T. Drummond. “Machine learning for high-speed corner detection”. In: *European Conference on Computer Vision*. Vol. 1. 2006, 430–Ð443.

[85] E. Adelson and J. Movshon. "Phenomenal coherence of moving visual patterns". In: *Nature* 200.5892 (Dec. 1982), pp. 523–525.

[86] J.A. Noble. "Finding corners". In: *Image Vision Computing* 6.2 (1988), pp. 121–128.

[87] Andrzej Cichocki and Rolf Unbehauen. "Neural networks for solving systems of linear equations and related problems". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 39.2 (1992), pp. 124–138.

[88] Steve Furber et al. "Overview of the spinnaker system architecture". In: *IEEE Transactions on Computers* 62.12 (Dec. 2013), pp. 2454–2467.

[89] Nicol N Schraudolph. "A fast, compact approximation of the exponential function". In: *Neural Computation* 11.4 (1999), pp. 853–862.

[90] Gavin C Cawley. "On a fast, compact approximation of the exponential function". In: *Neural computation* 12.9 (2000), pp. 2009–2012.

[91] Thomas B. Moeslund, Adrian Hilton, and Volker Kruger. "A survey of advances in vision-based human motion capture and analysis". In: *Computer Vision and Image Understanding*. Special Issue on Modeling People: Vision-based understanding of a person' s shape, appearance, movement and behaviour 104.2–3 (Nov. 2006), pp. 90–126. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2006.08.002. URL: http://www.sciencedirect.com/science/article/pii/S1077314206001263 (visited on 05/11/2015).

[92] Ronald Poppe. "Vision-based Human Motion Analysis: An Overview". In: *Comput. Vis. Image Underst.* 108.1–2 (Oct. 2007), pp. 4–18. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2006.10.016. URL: http://dx.doi.org/10.1016/j.cviu.2006.10.016 (visited on 05/11/2015).

[93] Ronald Poppe. "A survey on vision-based human action recognition". In: *Image and vision computing* 28.6 (2010), pp. 976–990.

[94] Huiyu Zhou and Huosheng Hu. "Human motion tracking for rehabilitation's survey". In: *Biomedical Signal Processing and Control* 3.1 (2008), pp. 1–18.

[95] Jungong Han et al. "Enhanced Computer Vision with Microsoft Kinect Sensor: A Review". In: *IEEE Transactions on cybernetics* 43.5 (2013), pp. 1318–1334.

[96] J Panaıté et al. "An experimental study of the Kinect's depth sensor". In: *IEEE International Symposium on Robotic and Sensors Environment*. 2011.

[97] Camille Simon-Chane et al. "Event-based tone mapping for Asynchronous Time-based Image Sensor". In: *Frontiers in Neuroscience; Neuromorphic Engineering* (2016).

[98] Julia A Lasserre, Christopher M Bishop, and Thomas P Minka. "Principled hybrids of generative and discriminative models". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 1. IEEE. 2006, pp. 87–94.

[99] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, 2008.

[100] Rizwan Chaudhry et al. "Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 1932–1939.

[101] Yoav Freund, Robert E Schapire, et al. "Experiments with a new boosting algorithm". In: *Icml*. Vol. 96. 1996, pp. 148–156.

[102] Robert E Schapire et al. "Boosting the margin: A new explanation for the effectiveness of voting methods". In: *Annals of statistics* (1998), pp. 1651–1686.

[103] Alex D Holub, Max Welling, and Pietro Perona. "Combining generative models and fisher kernels for object recognition". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 1. IEEE. 2005, pp. 136–143.

[104] Pablo Negri et al. "A cascade of boosted generative and discriminative classifiers for vehicle detection". In: *EURASIP Journal on Advances in Signal Processing* 2008 (2008), p. 136.

[105] Xiao Wang, Xavier Clady, and Consuelo Granata. "A human detection system for proxemics interaction". In: *Proceedings of the 6th international conference on Human-robot interaction*. ACM. 2011, pp. 285–286.

[106] Lionel Prevost et al. "Hybrid generative/discriminative classifier for unconstrained character recognition". In: *Pattern Recognition Letters* 26.12 (2005), pp. 1840–1848.

[107] Helmut Grabner, Peter M Roth, and Horst Bischof. "Eigenboosting: Combining discriminative and generative information". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[108] Thomas Deselaers, Georg Heigold, and Hermann Ney. "SVMs, Gaussian mixtures, and their generative/discriminative fusion". In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. IEEE. 2008, pp. 1–4.

[109] Jinjun Wang et al. "Discriminative and generative vocabulary tree: With application to vein image authentication and recognition". In: *Image and Vision Computing* 34 (2015), pp. 51–62.

[110] Yushi Jing, Vladimir Pavlović, and James M Rehg. "Boosted Bayesian network classifiers". In: *Machine Learning* 73.2 (2008), pp. 155–184.

[111] Xavier Lagorce and Ryad Benosman. "Stick: Spike time interval computational kernel, a framework for general purpose computation using neurons, precise timing, delays, and synchrony". In: *Neural computation* 27.11 (2015), pp. 2261–2317.

[112] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[113] Martin Lades et al. "Distortion invariant object recognition in the dynamic link architecture". In: *IEEE Transactions on computers* 42.3 (1993), pp. 300–311.

[114] Anil K Jain, Nalini K Ratha, and Sridhar Lakshmanan. "Object detection using Gabor filters". In: *Pattern Recognition* 30.2 (1997), pp. 295–309.

[115] Michael Lyons et al. "Coding facial expressions with gabor wavelets". In: *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. IEEE. 1998, pp. 200–205.

[116] Aapo Hyvarinen, Jarmo Hurri, and Patrick O. Hoyer. *Natural Image Statistics: a probabilistic approach to early computational vision*. Springer, 2009.

[117] Thusitha N Chandrapala and Bertram E Shi. "Invariant feature extraction from event based stimuli". In: *arXiv preprint arXiv:1604.04327* (2016).

[118] Claudette Cedras and Mubarak Shah. "Motion-based recognition: a survey". In: *Image and Vision Computing* 13.2 (1995), pp. 129–155.

[119] Md. Atique Rahman Abad et al. "Motion history image: its variants and applications". In: *Machine Vision and Applications* (2010).

[120] Jean-Matthieu Maro et al. "Event-based Visual Gesture Recognition with Background Suppression running on a smart-phone". In: *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*. IEEE. 2019, pp. 1–1.

[121] Garrick Orchard et al. "Converting static image datasets to spiking neuromorphic datasets using saccades". In: *Frontiers in neuroscience* 9 (2015), p. 437.

[122] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. "Poker-DVS and MNIST-DVS. Their history, how they were made, and other details". In: *Frontiers in neuroscience* 9 (2015), p. 481.

[123] Amos Sironi et al. "HATS: Histograms of averaged time surfaces for robust event-based object classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1731–1740.

[124] Yin Bi et al. "Graph-Based Object Classification for Neuromorphic Vision Sensing". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 491–501.

[125] Pramod Kumar Pisharady and Martin Saerbeck. "Recent methods and databases in vision-based hand gesture recognition: A review". In: *Computer Vision and Image Understanding* 141 (2015), pp. 152–165.

[126] Maryam Asadi-Aghbolaghi et al. "A survey on deep learning based approaches for action and gesture recognition in image sequences". In: *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*. IEEE. 2017, pp. 476–483.

[127] Kunwar Aditya et al. "Recent Trends in HCI: A survey on Data Glove, LEAP Motion and Microsoft Kinect". In: *2018 IEEE International Conference on System, Computation, Automation and Networking, ICSCA 2018* (2018), pp. 1–5. DOI: 10.1109/ICSCAN.2018.8541163.

[128] Zhengjie Wang et al. "Hand Gesture Recognition Based on Active Ultrasonic Sensing of Smartphone: A Survey". In: *IEEE Access* 7 (2019), pp. 111897–111922.

[129] Thomas Deselaers et al. "GyroPen: Gyroscopes for pen-input with mobile phones". In: *IEEE Transactions on Human-Machine Systems* 45.2 (2015), pp. 263–271. ISSN: 21682291. DOI: 10.1109/THMS.2014.2365723. URL: https://ieeexplore.ieee.org/abstract/document/6975206/.

[130] Hari Prabhat Gupta et al. "A Continuous Hand Gestures Recognition Technique for Human-Machine Interaction Using Accelerometer and Gyroscope Sensors". In: *IEEE Sensors Journal* (2016). ISSN: 1530437X. DOI: 10.1109/JSEN.2016.2581023.

[131] Chunyu Xie et al. *Deep Fisher Discriminant Learning for Mobile Hand Gesture Recognition*. Tech. rep. 2017. arXiv: 1707.03692v1.

[132] Heng-Tze Cheng et al. "Contactless gesture recognition system using proximity sensors". In: *Consumer Electronics (ICCE), 2011 IEEE International Conference on*. IEEE. 2011, pp. 149–150.

[133] Eun Ji Kim and Tae Ho Kang. *Mobile device having proximity sensor and gesture based user interface method thereof*. US Patent App. 12/814,809. June 2010.

[134] Jae Yeon Won et al. "Proximity sensing based on a dynamic vision sensor for mobile devices". In: *IEEE Transactions on Industrial Electronics* 62.1 (2015), pp. 536–544. ISSN: 02780046. DOI: 10.1109/TIE.2014.2334667.

[135] Lih Jen Kau et al. "A real-time portable sign language translation system". In: *Midwest Symposium on Circuits and Systems*. 2015. ISBN: 9781467365574. DOI: 10.1109/MWSCAS.2015.7282137.

[136] Bryce Kellogg, Vamsi Talla, and Shyamnath Gollakota. "Bringing Gesture Recognition to All Devices." In: *NSDI*. Vol. 14. 2014, pp. 303–316.

[137] Tomás Vega Gálvez et al. "Byte.it: Discreet teeth gestures for mobile device interaction". In: *Conference on Human Factors in Computing Systems - Proceedings* (2019), pp. 1–6. DOI: 10.1145/3290607.3312925.

[138] Biplab Ketan Chakraborty et al. *Review of constraints on vision-based gesture recognition for human-computer interaction*. 2018. DOI: 10.1049/iet-cvi.2017.0052. URL: https://doi.org/10.1049/iet-cvi.2017.0052.

[139] Houssem Lahiani, Monji Kherallah, and Mahmoud Neji. "Vision based hand gesture recognition for mobile devices: A review". In: *Advances in Intelligent Systems and Computing* 552.His (2017), pp. 308–318. ISSN: 21945357. DOI: 10.1007/978-3-319-52941-7_31.

[140] Sakher Ghanem, Christopher Conly, and Vassilis Athitsos. "A survey on sign language recognition using smartphones". In: *ACM International Conference Proceeding Series*. 2017. ISBN: 9781450352277. DOI: 10.1145/3056540.3056549. URL: http://dx.doi.org/10.1145/3056540.3056549.

[141] G. Ananth Rao and P. V.V. Kishore. "Sign language recognition system simulated for video captured with smart phone front camera". In: *International Journal of Electrical and Computer Engineering* (2016). ISSN: 20888708. DOI: 10.11591/ijece.v6i5.11384.

[142] Bryan G. Dadiz, John Michael B. Abrasia, and Jeezelle L. Jimenez. In: *2017 IEEE 2nd International Conference on Signal and Image Processing, ICSIP 2017*. Vol. 2017-Janua. Institute of Electrical and Electronics Engineers Inc., Nov. 2017, pp. 30–34. ISBN: 9781538609682. DOI: 10.1109/SIPROCESS.2017.8124500.

[143] Cheok Ming Jin, Zaid Omar, and Mohamed Hisham Jaward. "A mobile application of American sign language translation via image processing algorithms". In: *Proceedings - 2016 IEEE Region 10 Symposium, TENSYMP 2016*. 2016. ISBN: 9781509009312. DOI: 10.1109/TENCONSpring.2016.7519386.

[144] Houssem Lahiani, Mohamed Elleuch, and Monji Kherallah. "Real time hand gesture recognition system for android devices". In: *International Conference on Intelligent Systems Design and Applications, ISDA*. 2016. ISBN: 9781467387095. DOI: 10.1109/ISDA.2015.7489184.

[145] Ryad Benosman et al. "Event-Based Visual Flow". In: *IEEE Trans. Neural Netw. Learning Syst.* (2014).

[146] Rafael Serrano-Gotarredona et al. "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* (2009).

[147] Z. Ni et al. "Asynchronous Event-based Visual Shape Tracking for Stable Haptic Feedback in Microrobotics". In: *IEEE Transactions on Robotics* (2012).

[148] David Reverter Valeiras et al. "An Asynchronous Neuromorphic Event-Driven Visual Part-Based Shape Tracking". In: *IEEE transactions on neural networks and learning systems* (2015).

[149] Sadique Sheik et al. "Spatio-temporal spike pattern classification in neuromorphic systems". In: *Biomimetic and Biohybrid Systems*. Springer, 2013.

[150] Xavier Lagorce et al. "Spatiotemporal features for asynchronous event-based data". In: *Frontiers in neuroscience* (2015).

[151] Garrick Orchard et al. "HFirst: A Temporal Approach to Object Recognition". In: *TPAMI* (2015).

[152] Sio-Hoi Ieng et al. "Neuromorphic Event-Based Generalized Time-Based Stereovision". In: *Frontiers in Neuroscience* 12 (2018), p. 442. ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00442. URL: https://www.frontiersin.org/article/10.3389/fnins.2018.00442.

[153] David Reverter Valeiras et al. “Neuromorphic event-based 3d pose estimation”. In: *Frontiers in neuroscience* 9 (2016), p. 522.

[154] Jürgen Kogler, Christoph Sulzbachner, and Wilfried Kubinger. “Bio-inspired stereo vision system with silicon retina imagers”. In: *International Conference on Computer Vision Systems*. Springer. 2009, pp. 174–183.

[155] Elias Mueggler et al. “Lifetime estimation of events from dynamic vision sensors”. In: *2015 IEEE international conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 4874–4881.

[156] Henri Rebecq et al. “Events-to-video: Bringing modern computer vision to event cameras”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3857–3866.

[157] Bibrat Ranjan Pradhan et al. “N-HAR: A neuromorphic event-based human activity recognition system using memory surfaces”. In: *Proceedings - IEEE International Symposium on Circuits and Systems* 2019-May (2019). ISSN: 02714310. DOI: 10.1109/ISCAS.2019.8702581.

[158] Yuhuang Hu et al. “DVS benchmark datasets for object tracking, action recognition, and object recognition”. In: *Frontiers in neuroscience* 10 (2016), p. 405.

[159] Yanxiang Wang et al. “EV-Gait : Event-based Robust Gait Recognition using Dynamic Vision Sensors”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 6358–6367.

[160] A. Chadha et al. “Neuromorphic Vision Sensing for CNN-based Action Recognition”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (2019), pp. 7968–7972. ISSN: 15206149. DOI: 10.1109/ICASSP.2019.8683606.

[161] C. Cadena et al. “Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age”. In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[162] Guang Chen et al. “FLGR: Fixed Length Gists Representation Learning for RNN-HMM Hybrid-Based Neuromorphic Continuous Gesture Recognition”. In: *Frontiers in neuroscience* 13 (2019).

[163] Eun Yeong Ahn et al. “Dynamic vision sensor camera based bare hand gesture recognition”. In: *2011 IEEE Symposium On Computational Intelligence For Multimedia, Signal And Vision Processing*. IEEE. 2011, pp. 52–59.

[164] Junhaeng Lee et al. “Live demonstration: Gesture-based remote control using stereo pair of dynamic vision sensors”. In: *ISCAS 2012 - 2012 IEEE International Symposium on Circuits and Systems*. 2012. DOI: 10.1109/ISCAS.2012.6272144.

[165] Jun Haeng Lee et al. “Touchless hand gesture UI with instantaneous responses”. In: *Proceedings - International Conference on Image Processing, ICIP* (2012), pp. 1957–1960. ISSN: 15224880. DOI: 10.1109/ICIP.2012.6467270.

[166] Paul K.J. Park et al. "Gesture recognition system based on Adaptive Resonance Theory". In: *Proceedings - International Conference on Pattern Recognition*. 2012. ISBN: 9784990644109.

[167] Paul K.J. Park et al. "Computationally efficient, real-time motion recognition based on bio-inspired visual and cognitive processing". In: *Proceedings - International Conference on Image Processing, ICIP* 2015-Decem (2015), pp. 932–935. ISSN: 15224880. DOI: 10.1109/ICIP.2015.7350936.

[168] Bernhard Kohn et al. "Event-driven body motion analysis for real-time gesture recognition". In: *ISCAS 2012 - 2012 IEEE International Symposium on Circuits and Systems* (2012), pp. 703–706. DOI: 10.1109/ISCAS.2012.6272132.

[169] Kyoobin Lee et al. "Four DoF gesture recognition with an event-based image sensor". In: *1st IEEE Global Conference on Consumer Electronics 2012, GCCE 2012* (2012), pp. 293–294. DOI: 10.1109/GCCE.2012.6379606.

[170] Xavier Clady et al. "A Motion-Based Feature for Event-Based Pattern Recognition". In: *Frontiers in neuroscience* 10 (2016).

[171] Qian Liu and Steve Furber. "Real-time recognition of dynamic hand postures on a neuromorphic system". In: *Artificial Neural Networks* (2015).

[172] Sumit Bam Shrestha and Garrick Orchard. "Slayer: Spike layer error reassignment in time". In: *Advances in Neural Information Processing Systems* 2018-Decem (2018), pp. 1412–1421. ISSN: 10495258. arXiv: 1810.08646.

[173] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. "Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE)". In: (2018). arXiv: 1811.10766. URL: http://arxiv.org/abs/1811.10766.

[174] Qinyi Wang et al. "Space-time event clouds for gesture recognition: From rgb cameras to event cameras". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 1826–1835.

[175] Miguel Rivera-Acosta et al. "American Sign Language Alphabet Recognition Using a Neuromorphic Sensor and an Artificial Neural Network". In: *Sensors* 17.10 (2017), p. 2176.

[176] Jia Li et al. "Adaptive temporal pooling for object detection using dynamic vision sensor". In: *British Machine Vision Conference 2017, BMVC 2017*. 2017. ISBN: 190172560X. DOI: 10.5244/c.31.40.

[177] Kyu Min Kyung et al. "Background elimination method in the event based vision sensor for dynamic environment". In: *Digest of Technical Papers - IEEE International Conference on Consumer Electronics* (2014), pp. 119–120. ISSN: 0747668X. DOI: 10.1109/ICCE.2014.6775934.

[178] A. Elgammal, D. Harwood, and L. Davis. "Non-parametric model for background subtraction." In: *European conference on computer vision* (2000), pp. 751–767.

[179] C. Stauffer and W. E. L. Grimson. "Adaptive background mixture models for real-time tracking." In: *IEEE Transactions on Image processing* 2 (1999).

[180] O. Barnich and M. Van Droogenbroeck. "Vibe: A universal background subtraction algorithm for video sequences." In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 20.6 (2011), pp. 1709–1724.

[181] N. M. Oliver, B. Rosario, and A. P. Pentland. "A bayesian computer vision system for modeling human interactions." In: *IEEE transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 831–843.

[182] Mohammadreza Babaee, Duc Tung Dinh, and Gerhard Rigoll. "A Deep Convolutional Neural Network for Video Sequence Background Subtraction". In: *Pattern Recogn.* 76.C (Apr. 2018), pp. 635–649. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2017.09.040. URL: https://doi.org/10.1016/j.patcog.2017.09.040.

[183] Jacques Kaiser et al. "Embodied Neuromorphic Vision with Event-Driven Random Backpropagation". In: (2019), pp. 1–8. arXiv: 1904.04805. URL: http://arxiv.org/abs/1904.04805.

[184] Gregor Lenz, Sio-Hoi Ieng, and Ryad Benosman. "Event-based Dynamic Face Detection and Tracking Based on Activity". In: *CoRR* abs/1803.10106 (2018). arXiv: 1803.10106. URL: http://arxiv.org/abs/1803.10106.

[185] Carver Mead. "Neuromorphic electronic systems". In: *Proceedings of the IEEE* 78 (Oct. 1990).

[186] Christoph Posch, Ryad Benosman, and Ralph Etienne-Cummings. "Giving Machines Humanlike eyes". In: *IEEE Spectrum* 52 (Dec. 2015), pp. 44–49. DOI: 10.1109/MSPEC.2015.7335800.

[187] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Proceedings of NIPS, IEEE, Neural Information Processing System Foundation* (Jan. 2012), pp. 1097–1105.

[188] Clément Farabet et al. "Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel ConvNets for visual processing". In: *Frontiers in neuroscience* 6 (2012), p. 32.

[189] R. Tapiador-Morales et al. "Neuromorphic LIF Row-by-Row Multi-convolution Processor for FPGA". In: *IEEE Transactions on Biomedical Circuits and Systems* 13.1 (2019), pp. 159–169.

[190] S. Moradi et al. "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)". In: *IEEE Transactions on Biomedical Circuits and Systems* 12.1 (2018), pp. 106–122.

[191] A. Khodamoradi and R. Kastner. "O(N)-Space Spatiotemporal Filter for Reducing Noise in Neuromorphic Vision Sensors". In: *IEEE Transactions on Emerging Topics in Computing* (2018), pp. 1–1. ISSN: 2168-6750. DOI: 10.1109/TETC.2017.2788865.

[192] Alessandro Aimar et al. "NullHop:A Flexible Convolutional Neural Network Accelerator Based on Sparse Representations of Feature Maps". In: *CoRR* abs/1706.0 (2017).

[193] Leon Chua. *The Chua Lectures: From Memristors and Cellular Nonlinear Networks to the Edge of Chaos: Volume I Memristors: New Circuit Element with Memory*. Apr. 2020. ISBN: 978-981-12-1538-4. DOI: 10.1142/11693-vol1.

[194] Jean-Matthieu Maro and Ryad Benosman. "Event-based Gesture Recognition with Dynamic Background Suppression using Smartphone Computational Capabilities". In: *arXiv preprint arXiv:1811.07802* (2018).

[195] Ryad Benosman et al. "Event-based visual flow". In: *IEEE transactions on neural networks and learning systems* 25.2 (2013), pp. 407–417.

[196] Xavier Lagorce et al. "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.

[197] Chen Haoyu et al. "Learning to Deblur and Generate High Frame Rate Video with an Event Camera". In: *arXiv preprint arXiv:2003.00847* (2020).

[198] Zhe Jiang et al. "Learning Event-Based Motion Deblurring". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3320–3329.

[199] Bodo Rückauer et al. "Closing the Accuracy Gap in an Event-Based Visual Recognition Task". In: *arXiv preprint arXiv:1906.08859* (2019).

[200] Hanme Kim et al. "Simultaneous Mosaicing and Tracking with an Event Camera". In: Jan. 2014, pp. 1–12.

[201] Patrick Bardow, Andrew Davison, and Stefan Leutenegger. "Simultaneous Optical Flow and Intensity Estimation from an Event Camera". In: June 2016, pp. 884–892. DOI: 10.1109/CVPR.2016.102.

[202] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. "Asynchronous Spatial Image Convolutions for Event Cameras". In: *IEEERobot.Autom.Lett.*, 4 (Jan. 2019), pp. 816–822. DOI: 10.1109/LRA.2019.2893427.

[203] Christian Reinbacher, Gottfried Munda, and Thomas Pock. "Real-Time Intensity-Image Reconstruction for Event Cameras Using Manifold Regularisation". In: *International Journal of Computer Vision* (July 2016). DOI: 10.1007/s11263-018-1106-2.

[204] Henri Rebecq et al. "Events-To-Video: Bringing Modern Computer Vision to Event Cameras". In: June 2019, pp. 3852–3861. DOI: 10.1109/CVPR.2019.00398.

[205] Ana Maqueda et al. "Event-based Vision meets Deep Learning on Steering Prediction for Self-driving Cars". In: *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (June 2018).

[206] Rohan Ghosh et al. "Real-time object recognition and orientation estimation using an event-based camera and CNN". In: *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*. IEEE. 2014, pp. 544–547.

[207] Hongjie Liu et al. "Combined frame-and event-based detection and tracking". In: *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2016, pp. 2511–2514.

[208] Antoni Rosinol Vidal et al. "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 994–1001.

[209] A. Krikelis and C. C. Weems (editors). *Associative Processing and Processors*. 1997. ISBN: 978-981-12-1538-4. DOI: 10.1142/11693-vol1.

[210] Eugenio Culurciello, Ralph Etienne-Cummings, and Kwabena Boahen. "Arbitrated address-event representation digital image sensor". In: *Electronics Letters* 37 (Dec. 2001), pp. 1443–1445. DOI: 10.1049/el:20010969.

[211] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. "Poker-DVS and MNIST-DVS. Their History, How They Were Made, and Other Details". In: *Frontiers in Neuroscience* 9 (2015), p. 481. ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00481. URL: https://www.frontiersin.org/article/10.3389/fnins.2015.00481.

[212] Steven Dalton et al. "Optimizing Sparse Matrix Operations on GPUs Using Merge Path". In: May 2015, pp. 407–416. DOI: 10.1109/IPDPS.2015.98.

[213] Kostas Pagiamtzis and A. Sheikholeslami. "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey". In: *Solid-State Circuits, IEEE Journal of* 41 (Apr. 2006), pp. 712–727. DOI: 10.1109/JSSC.2005.864128.

[214] Zhe Yao, Vincent Gripon, and Michael Rabbat. "A Massively Parallel Associative Memory Based on Sparse Neural Networks". In: *arXiv:1303.7032* (Mar. 2013).

[215] Saeed Afshar et al. "Investigation of event-based memory surfaces for high-speed tracking, unsupervised feature extraction and object recognition". In: *Frontiers in Neuroscience* 12 (Mar. 2016). DOI: 10.3389/fnins.2018.01047.

[216] C. Brandli et al. "A 240 180 130 dB 3 s Latency Global Shutter Spatiotemporal Vision Sensor". In: *IEEE Journal of Solid-State Circuits* 49.10 (2014), pp. 2333–2341.

[217] A. Haron et al. "Parallel matrix multiplication on memristor-based computation-in-memory architecture". In: *2016 International Conference on High Performance Computing Simulation (HPCS)*. July 2016, pp. 759–766. DOI: 10.1109/HPCSim.2016.7568411.

[218] and Y. Kim and P. Li. "Architectural design exploration for neuromorphic processors with memristive synapses". In: *14th IEEE International Conference on Nanotechnology*. Aug. 2014, pp. 962–966. DOI: 10.1109/NANO.2014.6967962.

LIST OF FIGURES

LIST OF TABLES