HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition

Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E. Shi, *Fellow, IEEE*, and Ryad B. Benosman

Abstract—This paper describes novel event-based spatio-temporal features called time-surfaces and how they can be used to create a hierarchical event-based pattern recognition architecture. Unlike existing hierarchical architectures for pattern recognition, the presented model relies on a time oriented approach to extract spatio-temporal features from the asynchronously acquired dynamics of a visual scene. These dynamics are acquired using biologically inspired frameless asynchronous event-driven vision sensors. Similarly to cortical structures, subsequent layers in our hierarchy extract increasingly abstract features using increasingly large spatio-temporal windows. The central concept is to use the rich temporal information provided by events to create contexts in the form of time-surfaces which represent the recent temporal activity within a local spatial neighborhood. We demonstrate that this concept can robustly be used at all stages of an event-based hierarchical model. First layer feature units operate on groups of pixels, while subsequent layer feature units operate on the output of lower level feature units. We report results on a previously published 36 class character recognition task and a four class canonical dynamic card pip task, achieving near 100 percent accuracy on each. We introduce a new seven class moving face recognition task, achieving 79 percent accuracy.

Index Terms—Neuromorphic sensing, event-based vision, feature extraction

1 Introduction

Feature selection for object recognition is a fundamental problem in the study of visual processing. The open issue is always to determine how features of an image should be extracted and characterized. In traditional computer vision, visual features are typically defined as a function of the static luminance information within a local spatial neighborhood of an image [1], [2]. Different feature types differ only in the function they apply to the image. The temporal content of features has rarely been explored or tackled, mainly due to the three underlying hypotheses on which machine vision is based. The first hypothesis is that scenes are observed using a stroboscopic acquisition which produces a collection of static images (frames). Images are currently at the core of the whole field of artificial vision. So far, everything that has been developed has been designed to acquire, operate on, and display frames. A major drawback of frame-based acquisition is that it acquires information in a way that is independent of the dynamics of the underlying scene [3]. Scene illumination is measured at unnatural fixed time periods (frame-rate), resulting in acquisition of huge amounts of redundant data because most pixels will not change from one frame to the next. Massive redundancy in the acquired data is what allows video compression algorithms to achieve such impressive compression ratios (often around 50:1 [4]). However, before compression, this redundant data is still unnecessarily sampled, digitized, and transmitted, inducing a waste of resources, before expending even more resources to implement compression. This process sets important limitations on artificial perception that might one day be surmounted by using faster and more powerful computing devices (e.g., GPUs, clusters, etc.), but always at the cost of increasing power consumption. Nevertheless, the lack of dynamic content and the acquisition of both relevant and non-relevant data will always be the fundamental limit of images.

The second hypothesis of machine vision is that absolute pixel illumination (gray levels or colors) is the main source of information. However illumination is not an invariant property of a scene [1]. Most current algorithms fail to operate in uncontrolled lighting conditions. The ability to accurately measure luminance is also limited by the low dynamic range of conventional cameras [5].

The third hypothesis is that real-time operation implies a minimum of 24 images per second (the frame rate of common video formats [4]). There is currently a widespread belief in the field of artificial vision that high visual acquisition rates are only useful for cases where a fast changing stimulus must be observed. It is true that sensations of dynamic motion can be observed at 24 fps. However, it has been recently shown that biological retinas operate at temporal precision of 1 kHz (see [6]) because that is where most of the information of everyday scenes are [7]. If

- *X. Lagorce, F. Galluppi and R.B. Benosman are with the Vision and Natural Computation Group, Institut National de la Santé et de la Recherche Médicale, Paris F-75012, France Sorbonne Universités, Institut de la Vision, Université Paris 06, Paris F-75012, France, and with the Centre National de la Recherche Scientifique, Paris F-75012, France. E-mail: xavier.lagorce@ens-cachan.fr, francesco.galluppi@inserm.fr, ryad.benosman@upmc.fr.*
- *G. Orchard is with the Singapore Institute for Neurotechnology (SINAPSE), National University of Singapore, Singapore 119077. E-mail: garrickorchard@nus.edu.sg.*
- *B. Shi is with the Department of Electronic and Computer Engineering, and the Division of Biomedical Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong. E-mail: eebert@ust.hk.*

Manuscript received 13 Apr. 2015; revised 30 Apr. 2016; accepted 4 May 2016. Date of publication 8 July 2016; date of current version 12 June 2017. Recommended for acceptance by R. Collins.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2016.2574707

conventional scenes are processed at low temporal acquisition rates (30-60 Hz), it has been shown that there is a loss of 75 percent of valuable information leading to a poor separability between classes of objects [7]. Currently it is computationally and energetically expensive to process visual input in real time using conventional cameras at above 100 Hz. This because the amount of data which must be processed grows linearly with the frame rate, while the amount of information captured only grows sublinearly [7].

However, the field of Neuromorphic Engineering [8] has been developing bio-inspired event-driven, time-based vision sensors which operate on a very different principle [9]. Instead of capturing static images of the scene, these sensors record pixel intensity changes with high temporal precision. This high temporal precision provides information about scene dynamics which can aid in recognition and increase class separability [7]. In the last decade these sensors have matured to a point where they are now commercially available and can be operated by laymen. Event-driven time-based vision sensors promise to allow for power efficient low latency visual sensing in real world moving scenes, which has the potential for major impact in robotics, as well as mobile and wearable devices.

Event-driven time-based vision sensors provide data in the Address Event Representation (AER) format [10] which differs significantly from frames, and therefore conventional Machine Vision algorithms cannot be directly applied. For the task of object recognition, accuracy using event-driven time-based vision sensors still lags behind traditional approaches. Previous notable works on object recognition using event-driven time-based vision sensors include the Convolution AER Vision Architecture for Real-time (CAVIAR) project [11] which recognizes and tracks circles of different sizes using a hierarchical spiking network running on custom hardware. Later work progressed to differentiation between different shapes (circle, square, triangle) using an HMAX inspired algorithm also on custom silicon hardware [12]. Targeting more complex shapes, Perez-Carrasco et al. introduced a card pip recognition task which has been tackled in real-time using FPGAs running different hierarchical spiking models inspired by Convolutional Neural Networks (CNNs) [13] and HFirst [14].

Inspired by the popularity of the Mixed National Institute of Standards and Technology database (MNIST database) [2] in traditional machine vision, there has been a recent focus on character recognition using event-driven time-based vision sensors, which has been tackled using CNNs [13], Hierarchical like models [14], and Deep Belief Networks (DBNs) [15]. Perez Carrasco et al. showed how recent advances in training frame based CNNs can be leveraged by converting frame based CNNs to spiking CNNs for recognition [13], [16]. Moving past shape and character recognition, similar hierarchical models have been developed for application to human posture detection [17]. Frame-based CNNs have also been applied for discriminating between vehicles and pedestrians in a traffic scene, and recognizing household objects [18]. Other methods started to explore the integration of dynamical information into recognition by using motion-direction sensitive units [19] or dynamical networks (like Echo-state networks) [20].

This paper serves to advance the state of the art for performing recognition using event-driven time-based vision sensors. To provide comparison to previous works, we tackle the previously published card pip dataset [21] and character recognition tasks [14], achieving near perfect accuracy on both. As a first step towards performing human user recognition using event-driven time-based vision sensors, we introduce a new, more challenging, facial recognition task on which we achieve 79 percent accuracy, providing room for improvement in future work.

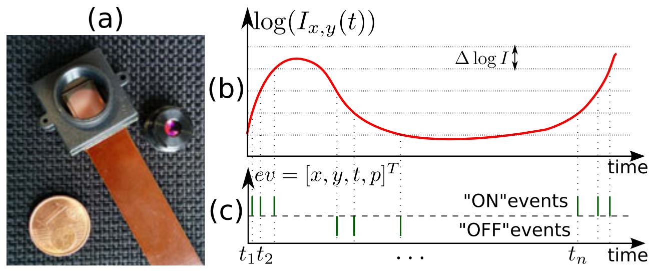


Fig. 1. Illustration of event-based encoding of visual signals. (a) ATIS camera [22]. (b) Log of the luminance of a pixel located at $[x, y]^T$. (c) Asynchronous temporal contrast events generated with respect to the predefined threshold $\Delta \log I$.

This paper begins with an introduction to the operation of event-driven time-based vision sensors in Section 2, followed by a description of the hierachical time-surface feature extraction technique in Section 3. We then describe performance of the technique in Section 4, before wrapping up with a discussion of results and a conclusion in Section 5 and Section 6.

2 Event-Driven Time-Based Vision Sensors

Biomimetic event-driven time-based vision sensors are a novel class of vision device that—like the biological retina—are driven by “events” happening within the visual scene. They are not like conventional vision sensors, which are driven by artificially created timing and control signals (e.g., frame clock) that have no relation whatsoever to the source of the visual information [23]. Over the past few years, a variety of these event-based devices has been developed, including temporal contrast vision sensors that are sensitive to relative luminance change [22], [23], [24], gradient-based sensors sensitive to static edges [25], and optical-flow sensors [26]. Most of these vision sensors output visual information about the scene in the form of asynchronous address events using the Address Event Representation protocol [10] and encode the visual information in the time dimension rather than as a voltage, charge, or current. The novel features we propose in this paper are designed to take advantage of the high temporal resolution data representation provided by event-based cameras, which is not provided by frame-based sensors.

This work uses datasets recorded by different event-based sensors [23], [27] and previously used in other publications [13], [14]. Due to the high accuracy we achieve on these dataset, we introduce a new set of recordings acquired using the Asynchronous Time-based Image Sensor (ATIS) [28]. This sensor is also a time-domain encoding vision sensor with 304×240 pixel resolution. The sensor contains an array of fully autonomous pixels that combine an illuminance relative change detector circuit and a conditional exposure measurement block.

The working principle of the ATIS sensor is shown Fig. 1. In this paper, we rely only on the output of a circuit contained in the ATIS pixels (see [22]) that detects relative changes in pixel log luminance over time. As soon as a change is detected, the process of communicating this change event off-chip is initiated. Off-chip communication executes with low latency (on the order of microseconds), ensuring that the time at which a change event is readout from the ATIS inherently represents the time at which the change was detected. This asynchronous low-latency readout scheme provides the high temporal resolution change detection data our features rely on. We discuss two types of change detection events: "ON" events and "OFF" events, which respectively indicate that an increase or decrease in log pixel intensity has been detected.

3 Model Description

In this section we describe the construction of our architecture for object recognition. We begin by formally defining time-surfaces (Section 3.1), and how time-surface prototypes can be learnt from input data (Section 3.2). Then we show how these time-surface prototypes can be arranged to form a hierarchical model (Section 3.3). Finally, in Section 3.4 we describe how classification is performed on the model output.

3.1 Time-Surface

The process of building a time-surface from the output of an event-driven time-based vision sensor is illustrated in Fig. 2 and described hereafter.

Consider a stream of visual events (Fig. 2b) which can be mathematically defined as

$$ev_i = [\mathbf{x_i}, t_i, p_i]^T, \quad i \in \mathbb{N}, \tag{1}$$

where ev_i is the ith event and consists of a location ($\mathbf{x_i} = [x_i, y_i]^T$), time ($t_i$) and polarity ($p_i$), with $p_i \in \{-1, 1\}$, where -1 and 1 represent OFF and ON events respectively. When an object (or the camera) moves, the pixels asynchronously generate events which form a spatio-temporal point cloud representing the object's spatial distribution and dynamical behavior. Fig. 2b shows such events generated by an object rotating in front of the sensor (Fig. 2a) where ON and OFF events are represented respectively by white and black dots.

Because the structure of this point cloud contains information about the object and its movement, we introduce the time-surface $\mathcal{S}_i$ of the ith event ev_i to keep track of the activity surrounding the spatial location $\mathbf{x_i}$ just before time t_i. We can then define $\mathcal{T}_i(\mathbf{u}, p)$ a time-context around an incoming event ev_i as the array of most recent events times at t_i for the pixels in the $(2R+1) \times (2R+1)$ square neighborhood centered at $\mathbf{x_i} = [x_i, y_i]^T$ as

$$\mathcal{T}_i(\mathbf{u}, p) = \max_{j \le i}\{t_j \,|\, \mathbf{x_j} = (\mathbf{x_i} + \mathbf{u}),\, p_j = p\}, \tag{2}$$

where $\mathbf{u} = [u_x, u_y]^T$ is such that $u_x \in \{-R, \ldots, R\}$, $u_y \in \{-R, \ldots, R\}$ and $p \in \{-1, 1\}\mathcal{T}_i(\mathbf{x}, p)$ is shown in Fig. 2d where intensity is coding for time values: bright pixels show recent activity whereas dark pixels received events further away in the past (only time values corresponding to OFF events are represented in the figure for clarity).

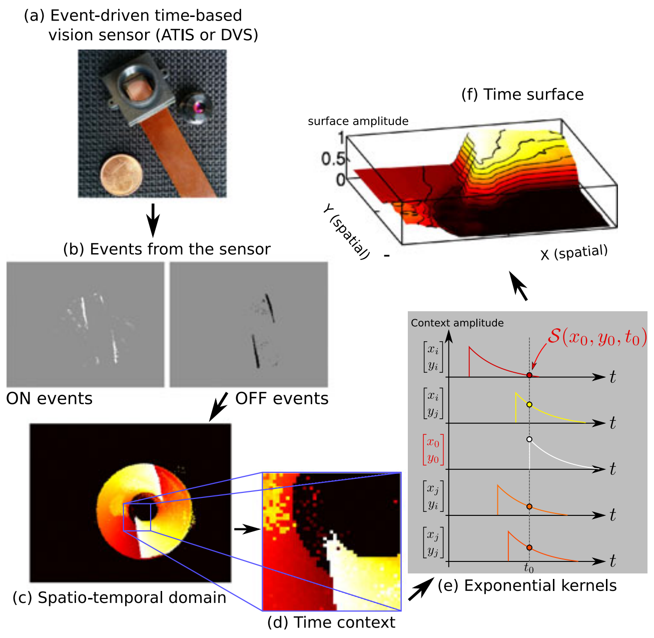


Fig. 2. Definition of a time-surface from the spatio-temporal cloud of events. A time-surface describes the recent time history of events in the spatial neighborhood of an event. This figure shows how the time-surface for an event happening at pixel $\mathbf{x_0} = [x_0, y_0]^T$ at time t_0 is computed. The event-driven time-based vision sensor (a) is filming a scene and outputs events shown in (b) where ON events are represented on the left hand picture and OFF events on the right hand one. For clarity, we continue by only showing values associated to OFF events. When an OFF event $ev_i = [\mathbf{x_0}, t_i, -1]$ arrives, we consider the times of most recent OFF events in the spatial neighborhood (c) where brighter pixels represent more recent events. Extracting a spatial receptive field allows to build the event-context $\mathcal{T}_i(\mathbf{x}, p)$ (d) associated with that event. Exponential decay kernels are then applyed to the obtained values (e) and their values at t_i constitute the time-surface itself. (f) shows these values as a surface. This representation will be used in the following figures and the label of the axes will be removed for better clarity.

Let $\mathcal{S}_i(\mathbf{u}, p)$, be the time-surface around an event ev_i (shown in Fig. 2e), it is defined by applying an exponential decay kernel with time constant τ on the values of $\mathcal{T}_i(\mathbf{u}, p)$

$$\mathcal{S}_i(\mathbf{u}, p) = e^{-(t_i - \mathcal{T}_i(\mathbf{u}, p))/\tau}. \tag{3}$$

$\mathcal{S}_i$ provides a dynamic spatiotemporal context around an event, the exponential decay expands the activity of passed events and provides information about the history of the activity in the neighborhood. The resulting surface $\mathcal{S}_i(\mathbf{u}, p)$ is shown in Fig. 2f for the OFF events represented all along Fig. 2. In the following sections $\mathcal{S}_i(\mathbf{u}, p)$ will be referred to directly as $\mathcal{S}_i$ to simplify notations. In the figures it will be represented as a surface showing the values of each of its element at their corresponding spatial positions.

Fig. 3 shows examples of time surfaces for simple moving edges. One can see, that a time surface is composed of two halves corresponding to the two polarity of incoming events. The first half has positive values, showing points corresponding to the ON events ($p = 1$) and the second half has negative values showing points corresponding to the OFF events ($p = -1$).

3.2 Learning Time-Surface Prototypes

Time-surface prototypes take the form of time-surfaces themselves, but whereas each incoming event will have a

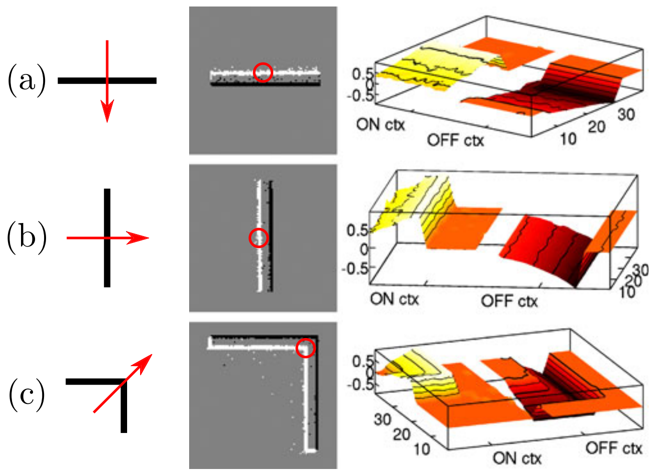


Fig. 3. Example of some time-surfaces for simple movements of objects. First column shows a representation of the stimulus. The second column shows corresponding data from the ATIS sensor where white dots are ON events and black dots are OFF events. The third column shows the time-surface obtained from these events and computed for the event located in the center of the circle in the second column: the first, positive, half is obtained from the ON events and the second, negative, half is obtained from the OFF events. (a) A horizontal bar moving downwards. (b) A vertical bar moving rightward. (c) Corner moving to the top-right.

different surface, the time-surface of each prototype remains constant (except during an initial learning phase). Time-surface prototypes are the set of elementary surfaces that are encountered in the observed scenes. The process of learning a time-surface for each prototype is described below, it relies on an incremental clustering process [29].

When an input event arrives at a bank of time-surface prototypes, the time-surface associated to the incoming event is calculated and compared to the time-surface of each prototype. The prototype with the time-surface most closely matching the surface of the input event will then generate an output event. We begin with a set of N inital time-surface prototypes, $\mathbf{C}_n$, $n \in [\![1, N]\!]$, where $\mathbf{C}_n$ takes the same form as $\mathcal{S}_i$ in (3). For initialization we simply use the first N time-surfaces as our initial values for the N prototypes. More formally

$$\mathbf{C}_n = \mathcal{S}_n \qquad n \in [\![1, N]\!]. \tag{4}$$

We then implement learning using the online clustering algorithm described in [29].

For each input event, ev_i, we calculate the time-surface, $\mathcal{S}_i$, and find $\mathbf{C}_k$, the time-surface prototype closest to $\mathcal{S}_i$ according to the euclidian distance. $\mathbf{C}_k$ is then updated using

$$\mathbf{C}_k \leftarrow \mathbf{C}_k + \alpha(\mathcal{S}_i - \beta \mathbf{C}_k), \tag{5}$$

with

$$\beta = \cos(\mathbf{C}_k, \mathcal{S}_i) = \frac{\mathbf{C}_k \cdot \mathcal{S}_i}{||\mathbf{C}_k|| \cdot ||\mathcal{S}_i||} \tag{6}$$

$$\alpha = \frac{0.01}{1 + \frac{p_k}{20000}}, \tag{7}$$

where p_k is the number of time-surfaces which have already been assigned to $\mathbf{C}_k$.

The full clustering process is summarized in Algorithm 1.

Algorithm 1. Online Clustering of Time-Surfaces

```
Ensure: $N$ cluster centers $\mathbf{C}_n$, $n \in [\![1, N]\!]$
  Use the first $N$ events' time-surfaces as initial values for $\mathbf{C}_n$, $n \in [\![1, N]\!]$
  Initialize $p_n \leftarrow 1$, $n \in [\![1, N]\!]$
  for every incoming event $ev_i$ do
    Compute time-surface $\mathcal{S}_i$
    Find closest cluster center $\mathbf{C}_k$
    $\alpha \leftarrow 0.01/(1 + p_k/20000)$
    $\beta \leftarrow \mathbf{C}_k \cdot \mathcal{S}_i/(||\mathbf{C}_k|| \cdot ||\mathcal{S}_i||)$
    $\mathbf{C}_k \leftarrow \mathbf{C}_k + \alpha(\mathcal{S} - \beta \mathbf{C}_k)$
    $p_k \leftarrow p_k + 1$
  end for
```

After this learning process, each time-surface can be associated to a particular prototype $\mathbf{C}_k$. In this manner, the stream of input events is transformed into a stream of prototype activations

$$feat_i = [x_i, y_i, t_i, k_i]^T, \tag{8}$$

where k_i is the index of the cluster center $\mathbf{C}_{k_i}$.

At this stage, when comparing an event's time-surface to a bank of time-surface prototypes one can also refine the process by adding some noise filtering. If an event is isolated (meaning that its time-surfaces is only made of a peak at the center), it can be dropped. It is also possible to add a maximum distance from the prototypes over which the time-surface is considered not to match any of the prototypes in the bank.

3.3 Creating a Hierarchical Model

Fig. 4 illustrates the hierarchical model we introduce in this paper. Steps (a) to (g) sum up the process described in the previous sections. As shown in Fig. 4, a moving digit (a) is presented to the ATIS camera (b) which produces ON and OFF events (c). Time-surfaces are built by convolving them with an exponential kernel of time constant τ_1 (d) and considering spatial receptive fields of sidelength $(2R_1 + 1)$. These time-surfaces are then clustered into N_1 prototypes represented as surfaces (e) in the Layer 1 box. When a cluster center is matched, an event is produced, resulting in the activations shown in (f). These events constitute the output of Layer 1 (g). One can see that each incoming event from the observed pattern is associated with the most representative prototype surface.

The nature of the output of Layer 1 is exactly the same as its input: Layer 1 outputs timed events. Once a prototype matches the temporal surface around the incoming event it immediately emits an event. Thus, the same steps used in Layer 1 (from (d) to (g)) can be applied in Layer 2. However the emitted event is now representing the temporal activity of a prototype surface, it thus carries more meaning than the initial camera event. The prototype surfaces of Layer 2 represent the temporal signature of the activity of complex features. Layer 2 uses different constants for space-time integration of features (R_2, N_2 and τ_2). The goal is to introduce stability of the perceptual representation and sensitivity to the accumulation of sensory evidence over time. This integration over longer and longer time period will thus be able to accumulate evidence in favor of alternative propositions in

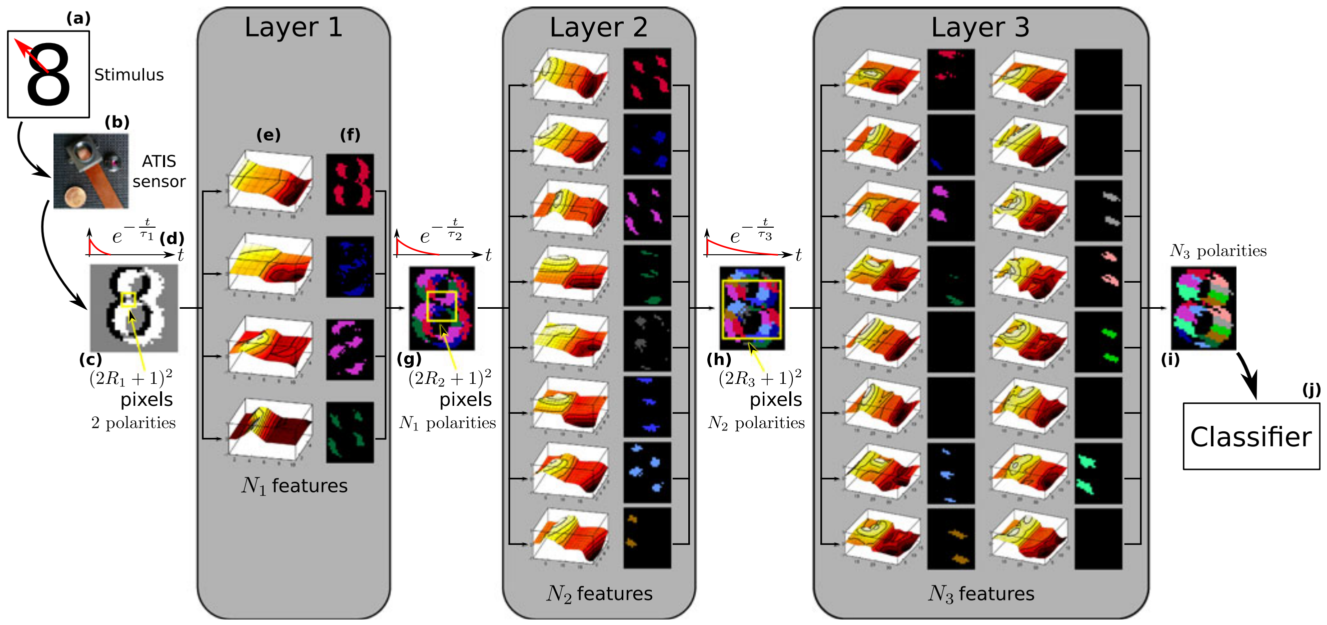


Fig. 4. View of the proposed hierarchical model. From left to right, a moving digit (a) is presented to the ATIS camera (b) which produces ON and OFF events (c) which are fed into Layer 1. The events are convolved with exponential kernels (d) to build event contexts from spatial receptive field of side-length $(2R_1 + 1)$. These contexts are clustered into N_1 features (e). When a feature is matched, it produces an event (f). Events from the N_1 features constitute the output of the layer (g). Each layer k (gray boxes) takes input from its previous layer and feeds the next by reproducing steps (d)-(g). The output of Layer k is presented between Layer k and $k+1$ ((g),(h),(i)). To compute event contexts, each layer considers a receptive field of side-length $(2R_k + 1)$ around each pixel. The event contexts are compared to the different features (represented as surfaces in the gray boxes as explained in Section 3.3) and the closest one is matched. The images next to each features show the activation of their associated features in each layer. These activations constitute the output of the layer. The output (i) of the last layer is then fed to the classifier (j) which will recognize the object.

a recognition process. When alternatives with a barely discernible difference in their sensory inputs are presented over an extended period of time, longer time and spatial integration scales can accumulate the small differences over time until it becomes eventually possible to discriminate the alternatives through its ever growing output. This accumulation dynamics is at the heart of the Hierarchy Of Time-Surfaces (HOTS) model, the difference between time scales can be substantial and can start from 50 ms for Layer 1 to 250 ms for Layer 2 to finally reach 1.25 s for Layer 3.

Layer 3 receives input from Layer 2, it is the last layer of the system and it provides the highest level information integration, as shown in Fig. 4i time-surface prototypes are also larger both spatially and temporally. The output of the temporal activity of Layer 3 can finally be used for object recognition by being fed to a classifier (shown in Fig. 4j).

As stated above, each layer is then defined by only a few parameters (we add an index l for the lth layer of the system):

- R_l, which defines the size of the time-surface neighborhood
- τ_l, the time constant of the exponential kernel applied to events
- N_l, the number of cluster centers (prototypes) learnt by the clustering algorithm.

To increase the information extracted by each subsequent layer, we make these parameters evolve between subsequent layer. For each layer, we define the parameters K_R, K_τ, K_N so that

$$R_{l+1} = K_R \cdot R_l \tag{9}$$

$$\tau_{l+1} = K_\tau \cdot \tau_l \tag{10}$$

$$N_{l+1} = K_N \cdot N_l. \tag{11}$$

The obtained architecture consists in a Hierarchy Of Time-Surfaces which is building and extracting a set of features (the prototypes from the final layer) out of a stream of input events. The time-surface prototypes will then be called time-surface features in the rest of the paper.

Fig. 3 shows what these features could be for the first layer of the achitecture where its input basis is constituted of only two vectors: ON events and OFF events. The other layers have input bases constituted of more vectors (as many as the number of features extracted by their previous layer), thus we could represent their features by a series of surfaces each corresponding to one feature of the previous layer. Because this representation is harder to relate to the actual input from the camera activating the feature, we chose to fuse these surfaces into their corresponding activity of ON and OFF events. The features of every layer of the architecture will then be represented as a set of two surfaces such as in Fig. 3, showing an image of the activity of ON and OFF events associated to the feature, this what is represented Fig. 4 in the gray boxes representing the different layers.

3.4 Classification

In this section we describe how the output of Layer 3 can be used as features for object recognition. Training of the recognition algorithm consists of two main steps. In the first step, different stimuli are presented to the model to learn the time-surface prototypes (referred to in the next sections as *features*) computed as described in the previous section.

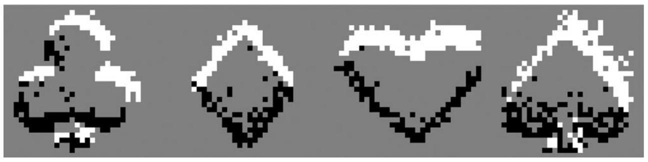


Fig. 5. **Flipped cards experiment:** Pattern database. The database for this experiment consists of the four suits (spades, hearts, clubs, and diamonds) found in a card deck. They are captured by a sensitive DVS sensor as the cards are flipped in front of it (white dots represent ON events and black dots OFF events).

This is the training phase of the algorithm (*model*). In a second step, the same learning stimuli are presented to the trained model and a histogram of the time-surface feature activations in the final layer is built for each object class. This is the training phases referred to as the *classifier*. A similar histogram is built for each test stimulus, it can then be compared to trained histograms to determine which object is present in the scene. The choice of the histogram is to show the robustness of the method, historams of activities as we will show are sufficient to provide reliable recognition scores. More complex classifier could be used specially time oriented ones such as Echo State Networks [30] or reccurent networks [31], these would allow the learning of the temporeal dynamics of activated features. However, as we will show in the experiment section, this is not necessary as the mean activity of features activation is sufficient to achieve high recognition scores.

When an object is presented to the camera between instants t_{start} and t_{end}, the time-surface feature activations form the set

$$\mathcal{F}(t_{start}, t_{end}) = \{feat_i \,|\, t_i \in [t_{start}, t_{end}]\}. \tag{12}$$

From this set, it is possible to build a histogram $\mathcal{H}$ counting how many times each feature has been activated, independently of its spatial position. This will constitute the signature of the observed objects.

To estimate the distance between two histograms, we use three different distances. We will refer to the standard distance when we will use the euclidian norm of the difference between two histograms (by looking at histograms as vectors in which the kth coordinate is the number of times feature k was matched)

$$d(\mathcal{H}_1, \mathcal{H}_2) = ||\mathcal{H}_1 - \mathcal{H}_2||. \tag{13}$$

We will refer to the normalized distance when we will use the euclidian norm of the difference between two histograms which have each been normalized by the number of generated features

$$d^N(\mathcal{H}_1, \mathcal{H}_2) = \left|\left|\frac{\mathcal{H}_1}{\mathrm{card}(\mathcal{H}_1)} - \frac{\mathcal{H}_2}{\mathrm{card}(\mathcal{H}_2)}\right|\right|, \tag{14}$$

where $\mathrm{card}(\mathcal{H}_k)$ is the total number of features counted in $\mathcal{H}_k$. We will also use the Bhattacharyya distance [32] defined as

$$d^B(\mathcal{H}_1, \mathcal{H}_2) = -\ln \sum_i \sqrt{\frac{\mathcal{H}_1(i)}{\mathrm{card}(\mathcal{H}_1)} \cdot \frac{\mathcal{H}_2(i)}{\mathrm{card}(\mathcal{H}_2)}}. \tag{15}$$

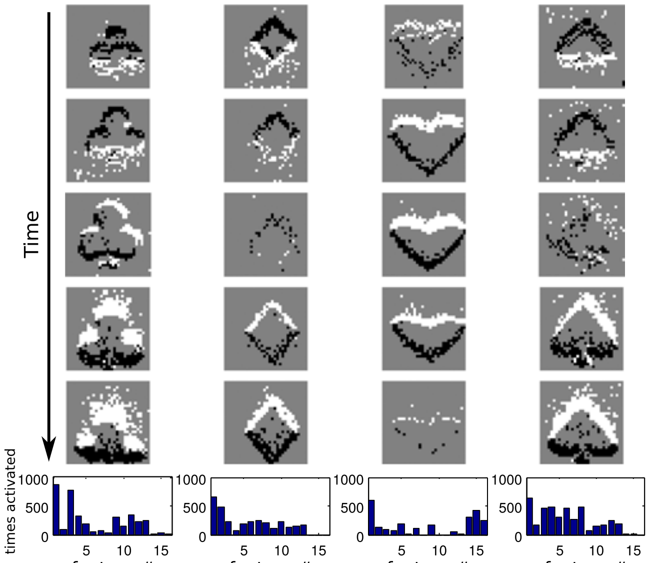


Fig. 6. **Flipped cards experiment:** Patterns' signatures. Histograms of feature activation numbers for the four suits moving in front of the camera. *X*-axis is the index of the feature shown in the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2574707, *Y*-axis is the number of activations of the feature during the stimulus presentation. Each column corresponds to one suit. The snapshots show how the pips evolve during one particular presentation (each snapshot is taken at a regular time interval). Each pattern outputs a different signature that allows its recognition.

Because these histograms are characteristics of the classes to recognize, we will refer to them as *signatures* in the rest of the paper.

4 Testing

The proposed method has been tested on three different tasks. The first consists of recognizing pips on poker cards as they are shuffled in front of the sensor to identify their suit. This task, which will be referred to as the *flipped card deck recognition task* (Section 4.1), has already been tackled by [13] and [14]. The second task is a simulated reading task in which characters are recognized as they move across the field of view of the sensor. This task, which will be referred as the *letters & digits recognition task* (Section 4.2), has already been tackled by [14]. These first two tasks have been chosen to provide comparison to previously published work. The third task is a face recognition task, which will be referred as the *Face recognition task* (Section 4.3). The data used for these different tasks are illustrated in Figs. 5, 8, and 12 respectively.

4.1 Flipped Card Deck Recognition Task

The first experiment is run on the card dataset provided by Teresa Serrano-Gotarredona and Bernabe Linares-Barranco [21] who captured the data using the sensitive DVS [27]. It represents a set of playing cards which are being flipped in front of the sensor (see Figs. 5 and 6). The pip representing the suit of the card has been isolated from the recording and the classifier has to determine the suit of the presented card. The data consists of ten presentations of each of the four suits, in the following, one presentation will be used for learning and the other nine for testing. Because the cards are being flipped by hand at high speed during the recording, an important deformation of the symbols occurs.

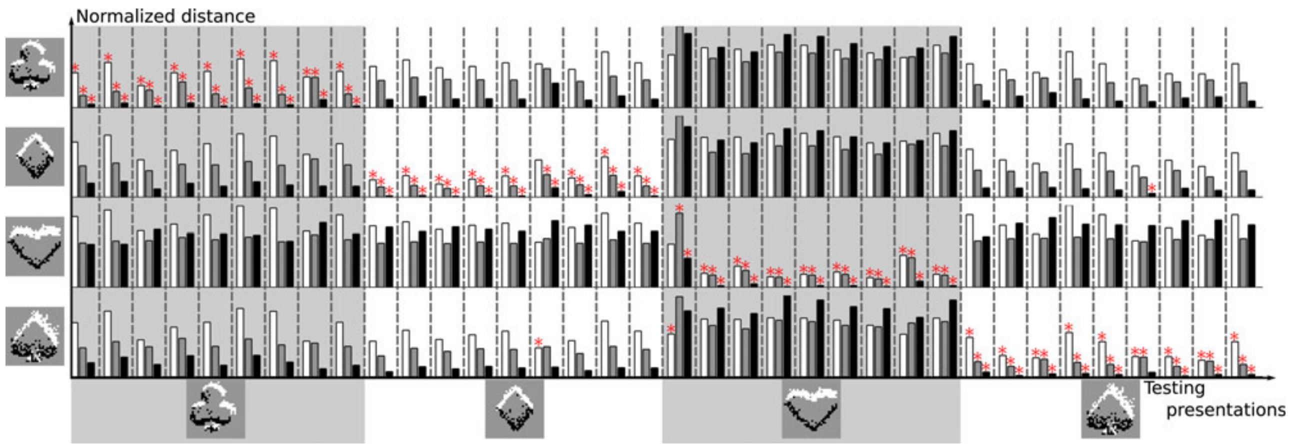


Fig. 7. **Flipped cards experiment:** Distance measurements between learning and testing presentations of patterns. The system is presented with nine different presentations of each learnt pattern. Each line presents data related to a particular trained pattern. Each section in between dashed vertical lines corresponds to the presentation of a test stimulus. Histograms show normalized distances obtained during the experiment so that the recognized objects is the smallest bar in each column (marked with a star). White bars code for standard distance, grey bars for normalized distance and black bars for Bhattacharyya distance. These three distances lead to respective performances of 94, 100 and 97 percent.

We use the hierarchical system described in the previous section with three layers. The parameters for the first layer are:

- $R_1 = 2$,
- $\tau_1 = 20\,\text{ms}$,
- $N_1 = 4$.

To go from one layer to the next, we use the following parameters:

- $K_R = 2$,
- $K_\tau = 10$,
- $K_N = 2$.

The sensor feeds spiking data into the first layer of the hierarchical model and histograms are built from the third layer's output. All four suits are used to train the model and each layer is trained sequentially. The features extracted in each layer are presented in the supplemental material, available online.

Training stimuli (a single presentation for each suit) are then presented to the system again to train the classifier. Examples of the classifier histograms are shown in Fig. 6. The rest of the stimulus examples (nine presentations) for each object are used for testing. The results can be seen in Fig. 7. In Fig. 7 the model and classifier have both been trained with only one presentation of each of the four suits. The nine other stimulus examples for each suit are used for testing.

The four rows along the vertical axis each show results for a different suit during testing. Each section between dashed lines shows histogram distances for one card flipped in front of the sensor. The different bars encode the histogram distances where white, gray, and black bars code for the standard, normalized, and Bhattacharyya distances respectively. The recognized class is the one with the smallest bar in each column, and is marked with a star. This particular experiment leads to performances of 94, 100 and 97 percent with the standard, normalized, and Bhattacharyya distances respectively.

Running some cross validation tests on the data gave us performances of 95-100 percent with all three distances.

4.2 Letters & Digits Recognition Task

The second experiment is run on a dataset provided by Orchard [14]. A DVS camera [23] is presented with the 10 digits from 0 to 9 and the 26 letters from the roman alphabet (see Fig. 8) printed on a barrel which was rotated at 40 rpm. The goal is to be able to classify correctly these 36 objects. This dataset consists of two presentations for each of the 36 patterns.

We use the same hierarchical system described previously with the same parameters as in the previous section.

The camera is feeding data into the first layer and the third layer's output is used by the classifier to recognize the letters and digits. Only the objects **F**, **V**, **O**, **8** and **B** are used to train the model. Each layer is trained sequentially. The trained features are available in the supplemental material, available online. Then, for each digit or character, we use one example to train the classifier. Some examples of the classifier histograms are provided in Fig. 9. All the signatures are also available in the supplemental material, available online. One testing presentation is then used for each object to test the classifier. The results can be seen Fig. 10. Each row along the vertical axis shows results for a different class during testing. Each section between dashed lines shows histogram distances for one character presented to the sensor. The different bars encode the histogram distances where white, gray, and black bars code for the standard,

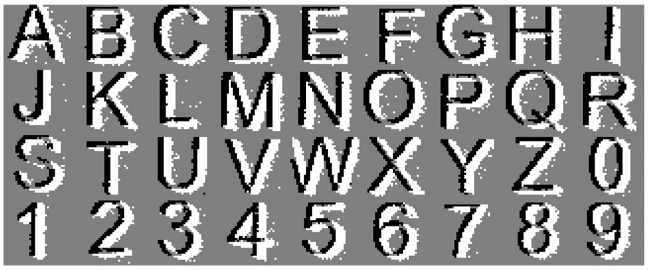


Fig. 8. **Letters & Digits experiment:** Pattern database. The database for this experiment consists of the 26 letters of the roman alphabet and the digits 0 to 9. They are captured by a DVS sensor as the characters are moving in front of it (white dots represent ON events and black dots OFF events).

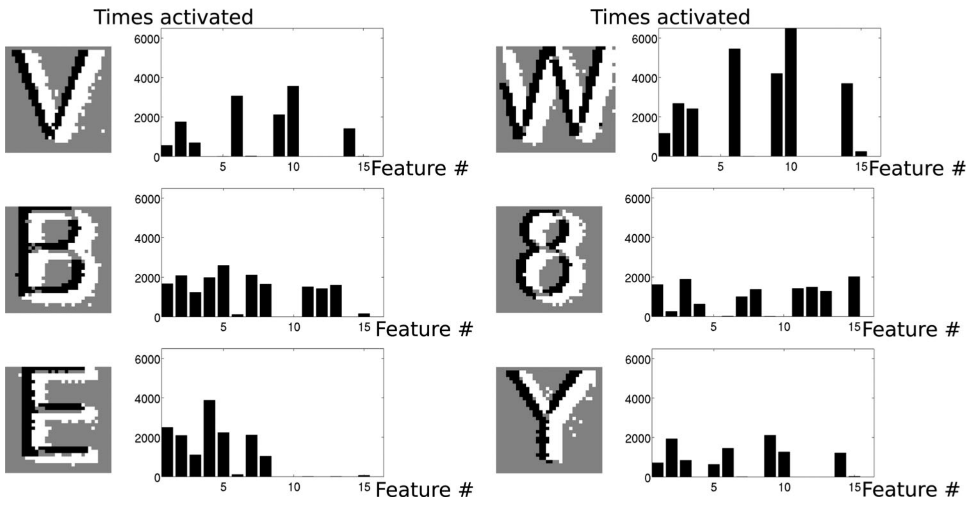


Fig. 9. **Letters & Digits experiment:** Pattern signatures for some of the input classes. For each letter and digit the trained histogram used as a signature by the classifier is shown. The snapshot shows an accumulation of events from the sensor (White dots for ON events and black dots for OFF events). The histograms present the signatures: *X*-axis is the index of the feature, *Y*-axis is the number of activations of the feature during the stimulus presentation. The signatures of all the letters & digits are presented in the supplemental material, available online.

normalized, and Bhattacharyya distances respectively. The recognized class is the one with the smallest bar in each column, and is marked with a star. Each character is presented once to the sensor in the order we gave to the classes so that perfect results correspond to stars only on the main diagonal. In this experiment, all distances lead to a recognition performance of 100 percent.

We ran a cross validation test by randomly choosing for each pattern which presentation would be used for learning (both the model and classifier), and the other is then used for testing. Every trial amongst the hundred we ran gave a recognition accuracy of 100 percent.

This experiment is run on a dataset composed of objects with very distinct spatial characteristics. Because of that, it is the best one to try to interpret the features learnt by the architecture. Fig. 11 presents the activation of all three layers' features for three different characters. We can observe on panels (a), (b) and (c) how the information available in different layers allow us to discriminate between three similar characters: **E**, **B** and **8**. Each column shows the response of a given feature (its associated surface representation is shown at the top) when the characters are presented (each line), with the last column showing all these data at once. We can see the difference in activation of the features corresponding to the differences in the input stimuli.

Panel (d) shows the accumulated feature activations for a set of objects used in this task. We can clearly see that the information encoded by each feature is becoming more and more abstract as we go from one layer to the next. In the second layer, features seem to respond to particular orientation of edges constituted of either ON or OFF events. In the third layer however, it seems that these features were pooled in order to recognize the line drawing the characters with features being tuned to its curvature. We can also see that the feature activations are very reproductible from one character to another containing the same inner shapes.

4.3 Face Recognition Task

The results obtained in the previous sections encouraged us to run the proposed method on more complex data. For our last experiment, we use a dataset consisting of the faces of seven subjects. Each subject is moving his or her head to follow the same visual stimulus tracing out a square path on a computer monitor (see Fig. 12). The data is acquired by an ATIS camera [22]. Each subject has 20 presentations in the dataset of which one is used for training and the others for testing.

We again use the same hierarchical system described previously with the following parameters:

- $R_1 = 4$,
- $\tau_1 = 50\,\text{ms}$,
- $N_1 = 8$.
- $K_R = 2$,
- $K_\tau = 5$,
- $K_N = 2$.

Because the faces are bigger and more complex objects, we use bigger receptive fields to define the time-surfaces and we set the layers to cluster twice as many features as in the previous experiments. These parameters lead to recognition performances of 37, 78 and 79 percent when using the standard, normalized and Bhattacharyya distances respectively as shown in Fig. 13.

5 Discussion

In this paper we have described a hierarchical architecture for object recognition using a new type of feature relying on *time-surfaces*. These time-surfaces use the high temporal resolution of event-driven time-based vision sensors to associate a descriptor to every event based on its relative timings to recent activity in its spatial neighborhood. The model then clusters this space to extract features. Successive layers perform this operation again and again, incorporating larger and larger spatial and temporal scales in the process. This allows for features of these successive layers to acquire more information, from bigger spatial receptive fields and from longer timescales.

Every layer also automatically learns its own features from its predecessor's output, removing the need for supervision. The process of training the model is thus completely unsupervised. Supervision only takes place when training

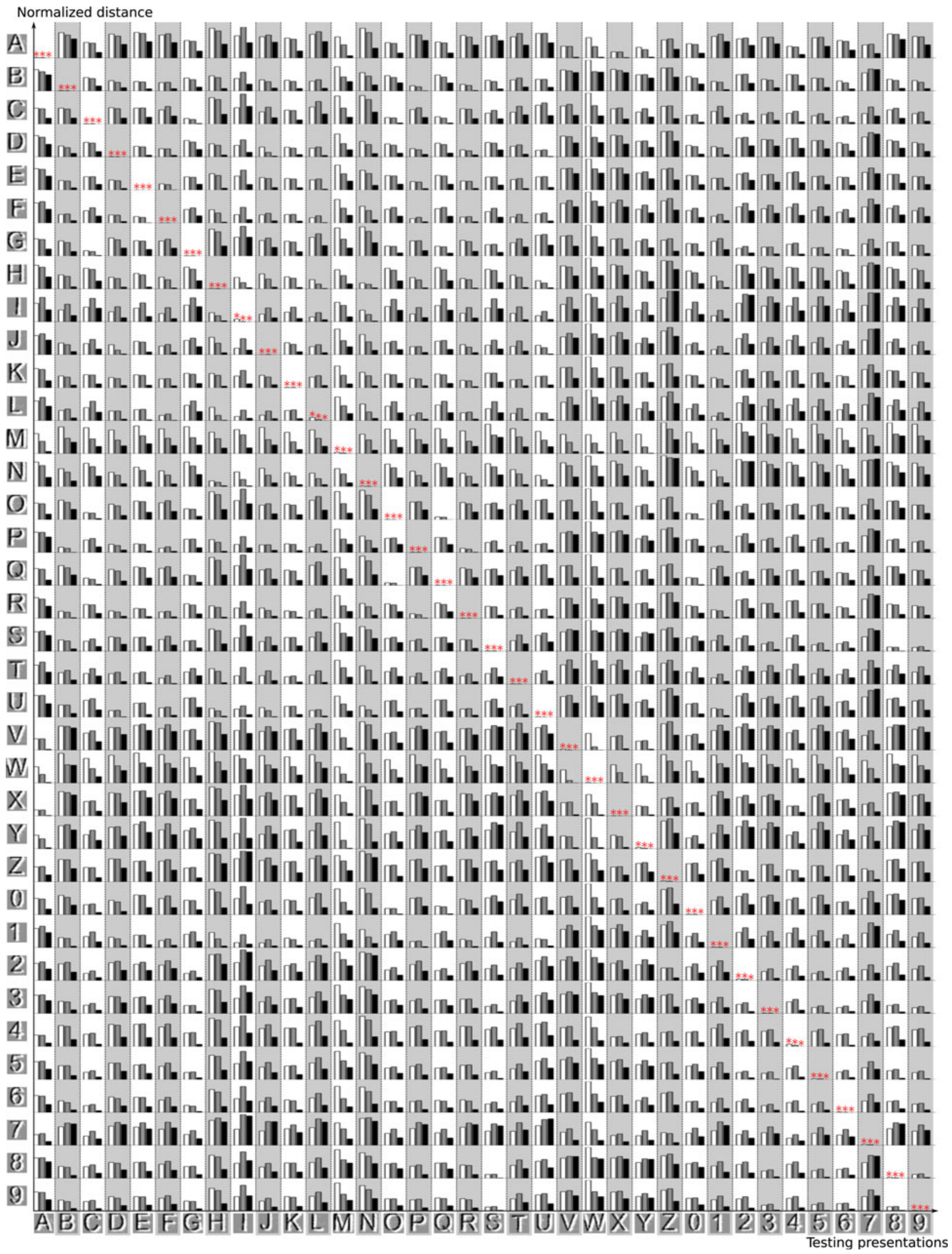


Fig. 10. **Letters & Digits experiment:** Distance measurements between learning and testing presentations of patterns. One presentation of each learnt pattern is shown to the system. Each line presents data related to a particular trained pattern. Each section in between dashed vertical lines corresponds to the presentation of a test stimulus. Histograms show normalized distances obtained during the experiment so that the recognized objects is the smallest bar in each column (marked with a star). White bars code for standard distance, grey bars for normalized distance and black bars for Bhattacharyya distance. These three distances all lead to a 100 percent recognition rate.

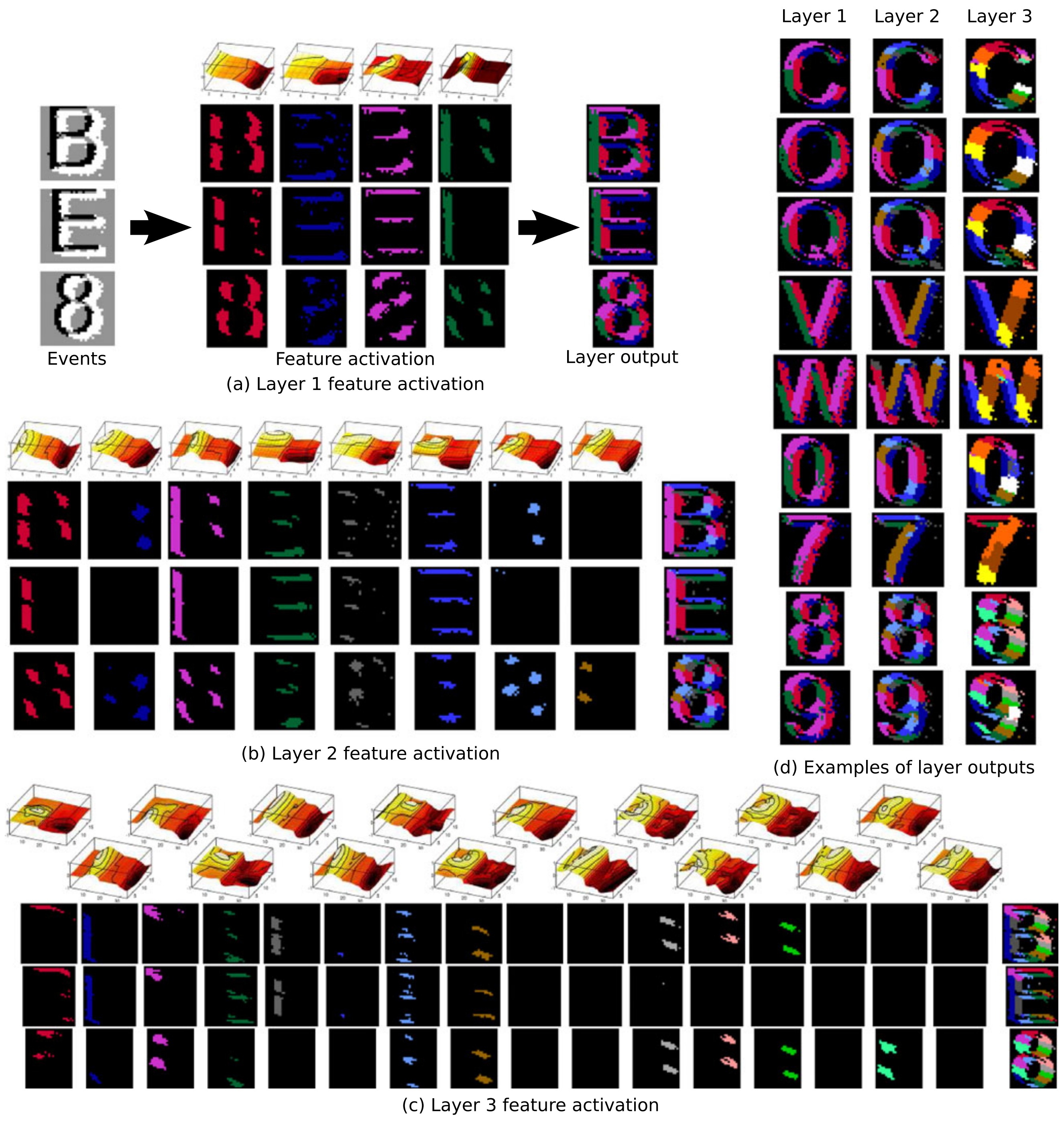


Fig. 11. **Letters & Digits experiment:** Activation of features in the different layers. (a), (b) and (c) activation of the different features extracted by each of the three layers for some chosen characters: **E**, **B** and **8**. The last column shows the superposition of the different features (each color corresponds to a particular feature). The other columns show the independent activation of the different features with their associated surface representation. Each of the three line of activations presents a different character.(a) shows the output of the first layer, (b) of the second and (c) of the third. Differences in the information offered by each layer to differentiate between these similar classes can be observed. (d) Examples of outputs of the three layers for a set of recognized objects. We can see how the information becomes more abstract when increasing layers, going from orientation of contours of a given polarity (ON or OFF events) to contours of a given curvature.

the classifier, which inherently requires knowledge of the object class.

Other object recognition methods such as HMAX [33] or HFirst [14] extract orientations as a first step using oriented Gabor filters, thus only looking at the spatial repartition of the input data or events to build features. HFirst does make use of time in computation, but it uses time to encode signal strength of spatial features rather than capturing scene dynamics. In contrast, time-surfaces use both spatial and temporal information to build features which are then not only spatial, but spatio-temporal features. Time-surfaces are, by design, encoding spatial information such as shapes and temporal information such as motion in the feature space. This makes them interesting features for recognition in dynamic environments. More than recognizing moving objects, these features should be able to extract the intrinsic relative inner movements of objects to help the recognition process. In the architecture described in this work, the

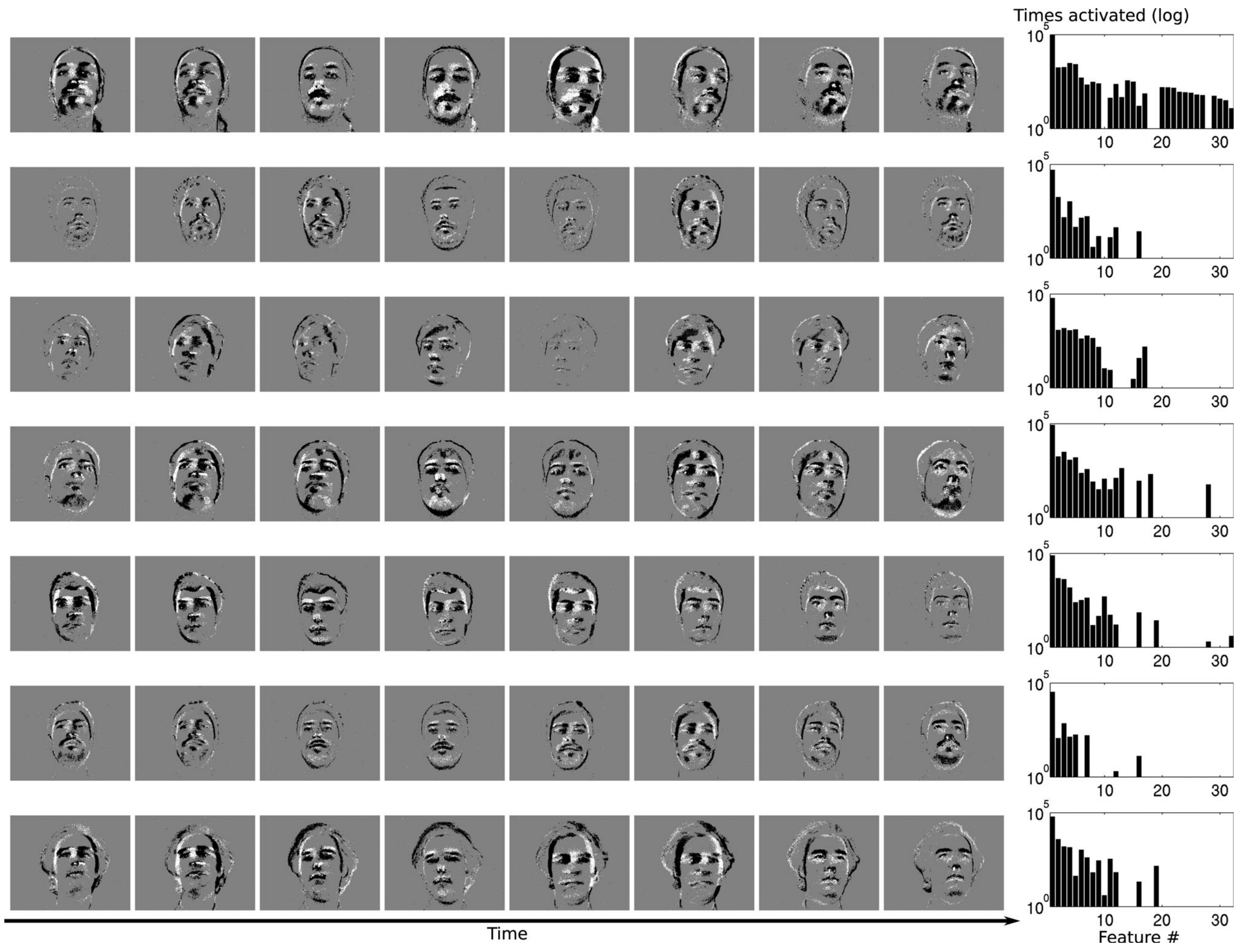


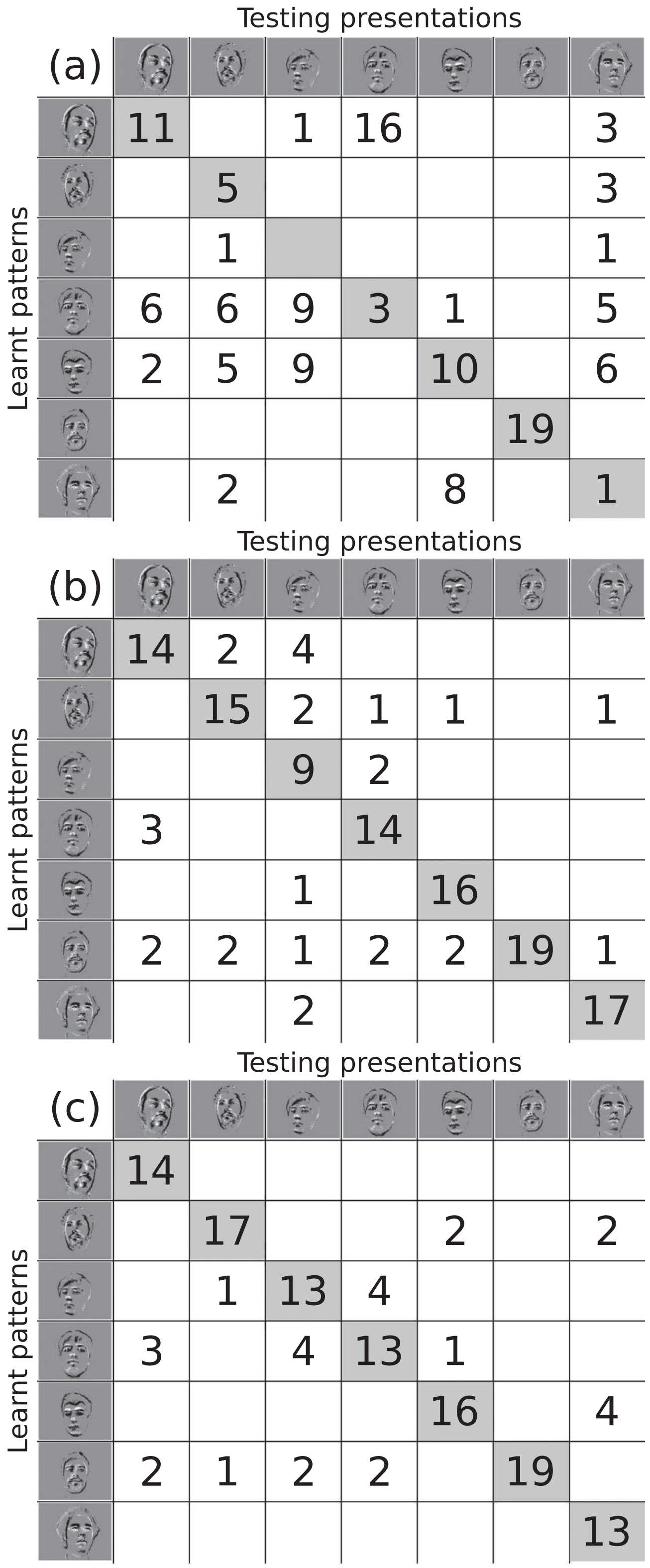
Fig. 12. **Faces recognition experiment:** Pattern database and signatures. The database is constituted of seven faces moving their gaze by following a dot in a square movement. The snapshots show the obtained stimulus with successive positions in time of the faces (white dots represent ON events and black dots represent OFF events; snapshots are taken every 350 ms.). The last column presents the signature learnt by the classifier for each face using the 32 features of the third layer of the system: *X*-axis is the index of the feature shown in the supplemental material, available online, *Y*-axis is the number of activations of the feature during the stimulus presentation (in logarithmic scaling).

extracted features are still very dependent from the direction and speed of motion of the objects we want to recognize. Thus for an object to be recognized independently from its motion, a lot of different movement patterns have to be presented to the system during learning. This could be overcome using basis function decomposition similarly to what has been introduced in [34]. This concept will be studied in an extension of this work.

Masquelier et al. [35] used STDP to learn features as event patterns. But STDP in its basic and most used form can only extract the co-activation of different input neurons and thus cannot extract the exact order of activation. Because time-surfaces encodes relative timings, they can distinguish between different orders of activation which is important for real spatio-temporal features. Moreover because time-surfaces are generated around every event, we also do not need to stabilize or track objects for their recognition and features will be recognized whatever their position in the field of view of the sensor.

The letters & digits experiment demonstrates the generalization strength of the feature extraction process. To recognize the 36 characters constituting the dataset, we only used a subset of five characters representative of the whole. When the different layers were learning features, they were only shown the characters **F**, **V**, **O**, **8** and **B**. Then, signatures were obtained for all of the 36 characters of the dataset to train the classifier stage. This still leads to a recognition performance of 100 percent which is showing us than the system can recognize objects without the need for its features to be specifically trained on those objects.

One can note that the parameters of the system still had to be tuned differently for the different experiments presented in the paper. Coping with the varying nature of the input stimulus can be achieved in different ways. In the architecture presented in this work, we use only one classifier fed by the last layer of the system, thus only operating recognition on the higher level features which are also the slowest (because they integrate information over the longest period of time). It is possible to add intermediate classifiers at the output of each layer in order to get different classifications results, each operating on features built over longer and longer period of time. Thus the first layers' classifier would give the fastest but most inaccurate results whereas the following layers would give more and more reliable results with a larger and larger detection time. Increasing the numbers of layers would then allow us to get classification



(a)

11		1	16			3
	5					3
	1					1
6	6	9	3	1		5
2	5	9		10		6
					19	
	2			8		1

(b)

14	2	4				
	15	2	1	1		1
		9	2			
3			14			
		1		16		
2	2	1	2	2	19	1
		2				17

(c)

14						
	17			2		2
	1	13	4			
3		4	13	1		
				16		4
2	1	2	2		19	
						13

Fig. 13. **Face recognition experiment:** Recognition results. Subfigure show the recognition results for the three different distances used. (a) Standard distance: 37 percent accuracy. (b) Normalized distance: 78 percent accuracy. (c) Bhattacharyya distance: 79 percent accuracy. The system is presented with 19 testing presentations of each learnt class. These 19 presentations are merged into the columns and the numbers are indicating how many matches are obtained for the different learnt patterns. Perferct results would fill the diagonal (gray background cells) with values 19, numbers in other cells correspond to classification errors.

TABLE 1
Comparison with State of the Art

Author	Cards dataset	Letters & Digits	Faces
Serrano-Gotarredona et al. [21]	$90.1\% - 91.6\%$	–	–
Orchard et al. [14]	**$97.5\% \pm 3.5\%$**	$84.9\% \pm 1.9\%$	–
Lagorce et al. (Hereby) *Standard distance*	**$95\% - 100\%$**	**100%**	37%
Lagorce et al. (Hereby) *Normalized distance*	**$95\% - 100\%$**	**100%**	**78%**
Lagorce et al. (Hereby) *Bhattacharyya distance*	**$95\% - 100\%$**	**100%**	**79%**

results computed from different timescales without further tuning of the system. Each layer could also compute time-surfaces with different time constants for their exponential kernels by adding an internal competition system (such as lateral inhibition or a winner-take-all circuit).

Recognizing a time-surface essentially consists of performing coincidence detection on input event arriving in a specific spatial and temporal pattern. For instance, this could be implemented by using leaky integrate and fire neurons and delay lines. Thus, the model described in this paper can be seen as a Spiking Neural Network and could be implemented on neuromorphic hardware for neural simulation. Moreover, every pixel is considering its own receptive field to build its associated time-surface. This can allow most of the processing to be parallelized to be implemented on platforms such as FPGAs or new, highly parallel, computer architectures such as SpiNNaker [36].

Several improvements can also be considered to improve the architecture proposed in this paper and are considered as possible futur work. In the current method, every incoming event generates an output event. But events both close in time and space generate similar time-surfaces. Inhibition and Winner-take-all mechanisms could be added to reduce the amount of redundancy introduced by these similarities as have been done in [20], [37]. Object recognition is also undertaken after the whole object has finished passing by. We chose to operate that way because of the final classifier that we used, which we wanted to keep as simple as possible to show that information from the network are valuable compared to other architectures. If another online classifier is used, such as an ELM [38] then we could get intermediate classification results as time-surface are arriving.

One can also note that even if our method is inspired by biology, the timescales that increase geometrically across the successive layers of the system are fairly large in the latest layers. This not compatible anymore with biological time constants. Our method is a proof of concept of a hierarchy of time scales that does not aim at explaining the brain. In principle though, information could be integrated over even longer timescales. We could then imagine large networks such as the ones used currently by Google [39] where the last layer could code for integration over months.

6 Conclusion

We have presented a hierarchical recognition architecture using a new way of representing features in the spatio-temporal output of asynchronous change detection vision sensors. These features are using relative timings of events, enabled by the high temporal precision of these sensors' output, to give contextual information to events, which we have called time-surfaces. The proposed architecture matches or improves the current state of the art on two previously published recognition tasks and results for a third, more difficult task have been presented. This comparison is summed-up in Table 1.

Acknowledgments

This work was performed in the frame of the LABEX LIFESENSES [ANR-10-LABX-65] and was supported by French state funds managed by the ANR within the Investissements d'Avenir programme [ANR-11-IDEX-0004-02]. XL has been supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 604102 (HBP). Franco-Singaporean collaboration on this project was supported by the Merlion Programme of the Institut Franais de Singapour, under administrative supervision of the French Ministry of Foreign Affairs and the National University of Singapore. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the French Ministry of Foreign Affairs. The authors would also like to acknowledge discussion at the US National Science Foundation Telluride Neuromorphic Cognition Engineering Workshop.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94
[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, " Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=726791
[3] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proc. IEEE*, vol. 102, no. 10, pp. 1470–1484, Oct. 2014.
[4] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Norwell, MA, USA: Kluwer, 1997.
[5] E. Reinhard, W. Heidrich, P. Debevec, S. Pattanaik, G. Ward, and K. Myszkowski, *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. San Francisco, CA, USA: Morgan Kaufmann, 2010.
[6] T. Gollisch and M. Meister, "Eye smarter than scientists believed: Neural computations in circuits of the retina," *Neuron*, vol. 65, no. 2, pp. 150–164, 2010.
[7] H. Akolkar, et al., "What can neuromorphic event-driven precise timing add to spike-based pattern recognition?" *Neural Comput.*, vol. 27, pp. 561–593, 2015.
[8] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.
[9] C. Mead and M. Mahowald, "The silicon retina," *Sci. Amer.*, vol. 264, pp. 76–82, 1991.
[10] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address-events," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 47, no. 5, pp. 416–434, May. 2000.
[11] R. Serrano-Gotarredona, et al., "CAVIAR: A 45k neuron, 5 M synapse, 12 G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1417–38, Sep. 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19635693
[12] F. Folowosele, R. J. Vogelstein, and R. Etienne-Cummings, "Towards a cortical prosthesis: Implementing a spike-based HMAX model of visual object recognition in silico," *IEEE J. Emerging Select. Topics Circuits Syst.*, vol. 1, no. 4, pp. 516–525, Dec. 2011. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6133300
[13] J. A. Pérez-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," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2706–19, Nov. 2013.
[14] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "Hfirst: A temporal approach to object recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 10, pp. 2028–2040, Oct. 2015.
[15] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers Neuroscience*, vol. 7, Jan. 2013, Art. no. 178.
[16] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *Int. J. Comput. Vis.*, vol. 113, no. 1, pp. 54–66, 2015.
[17] S. Chen, P. Akselrod, B. Zhao, J. A. P. Carrasco, B. Linares-Barranco, and E. Culurciello, " Efficient feedforward categorization of objects and human postures with address-event image sensors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 302–14, Feb. 2012.
[18] R. Ghosh, A. Mishra, G. Orchard, and V. Thakor, "Real-time object recognition and orientation estimation using an event-based camera and CNN," in *Proc. IEEE Biomed. Circuits. Syst. Conf.*, 2014, pp. 544–547.
[19] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, 2007, pp. 1–8.
[20] X. Lagorce, S.-H. Ieng, X. Clady, M. Pfeiffer, and R. B. Benosman, "Spatiotemporal features for asynchronous event-based data," *Frontiers Neuroscience*, vol. 9, 2015, Art. no. 46.
[21] T. Serrano-Gotarredona and B. Linares-Barranco, "Poker-DVS and MNIST-DVS. Their history, how they were made, and other details," *Frontiers Neuroscience*, vol. 9, 2015, Art. no. 481. [Online]. Available: http://www.frontiersin.org/neuromorphic_engineering/10.3389/fnins.2015.00481/abstract
[22] 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," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259 –275, Jan. 2011.
[23] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.
[24] J. A. Lenero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 3.6 μs latency asynchronous frame-free event-driven dynamic-vision-sensor," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1443–1455, Jun. 2011. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5746543
[25] A. G. Andreou and K. A. Boahen, "A contrast sensitive silicon retina with reciprocal synapses," in *Advances in Neural Information Processing Systems*. San Francisco, CA, USA: Morgan Kaufmann, 1991, pp. 764–772. [Online]. Available: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.8246
[26] A. A. Stocker, "Analog integrated 2-D optical flow sensor," *Analog Integr. Circuits Signal Process.*, vol. 46, no. 2, pp. 121–138, 2006.
[27] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128×128 1.5 percent contrast sensitivity 0.9 percent FPN 3 μs latency 4 mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, Mar. 2013.
[28] C. Posch, D. Matolin, and R. Wohlgenannt, "An asynchronous time-based image sensor," in *Proc. IEEE Int. Symp. Circuits Syst.*, May2008, pp. 2130 –2133.
[29] D. Ballard and J. Jehee, "Dynamic coding of signed quantities in cortical feedback circuits," *Frontiers Psychology*, vol. 3, 2012, Art. no. 254.
[30] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," GMD—German Nat. Res. Inst. Comput. Sci., Tech. Rep. GMD Report 148, 2001, http://www.faculty.jacobs-university.de/hjaeger/pubs/EchoStatesTechRep.pdf

[31] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for improved unconstrained handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009.

[32] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by probability distributions," *Bulletin Calcutta Math. Soc.*, vol. BCMS-35, pp. 99–109, 1943.

[33] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 411–426, Mar. 2007.

[34] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

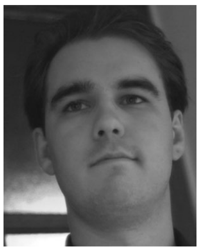
[35] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Comput. Biol.*, vol. 3, no. 2, 2007, Art. no. e31.

[36] S. Furber, F. Galluppi, S. Temple, and A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6750072

[37] X. Lagorce, S.-H. Ieng, and R. Benosman, "Event-based features for robotic vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 4214–4219.

[38] J. Tapson, et al., "Synthesis of neural networks for spatio-temporal spike pattern recognition and processing," *arXiv:1304.7118*, 2013, http://dx.doi.org/10.3389/fnins.2013.00153

[39] J. Dean, et al., "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.



Xavier Lagorce received the BSc degree, the Agrégation degree in electrical engineering, and the MSc degree in advanced systems and robotics from the École Normale Supérieure de Cachan, Cachan, France, in 2008, 2010 and 2011, respectively. He is currently working toward the PhD degree in neurally inspired hardware systems and sensors in the Vision Institute, Paris.

Garrick Orchard received the BSc degree in electrical engineering from the University of Cape Town, South Africa, in 2006 and the MSE and PhD degrees in electrical and computer engineering from Johns Hopkins University, Baltimore, in 2009 and 2012, respectively. He has been named a Paul V. Renoff fellow in 2007 and a Virginia and Edward M. Wysocki, senior fellow in 2011. He has received the JHUAPL Hart Prize for Best R&D Project in 2009, and the IEEE BioCAS best live demo award in 2012. He is currently a postdoctoral research fellow in the Singapore Institute for Neurotechnology (SINAPSE), National University of Singapore where his research focuses on developing neuromorphic vision sensors and algorithms for high speed vision tasks. His other research interests include mixed-signal VLSI, compressive sensing, navigation, and legged locomotion.

Francesco Galluppi received the BSc degree in electronic engineering from the University of Rome 'Roma Tre' in 2005, and the MSc degree in cognitive psychology from the University of Rome 'La Sapienza' in 2010. He received the PhD degree while working on neurally inspired hardware systems, sensors and robots from the University of Manchester in 2013. He is a postdoctoral research associate in the Institut de la Vision, University Pierre et Marie Curie in Paris. He visited the Departamento de Tecnologa Electrnica of the University of Malaga in 2007, working on robotic assistive technologies. He joined the SpiNNaker group at the University of Manchester. His interests lie in studying the biological basis of human behavior, and how technology interacts with it. His current research focus on the use event-based stimulation with assistive technologies for blind people.

Bertram E. Shi (S'93-M'95-SM'00-F'01) received the BS and MS degrees in electrical engineering from Stanford University, Stanford, CA, USA in 1987 and 1988, respectively, and the PhD degree in electrical engineering from the University of California, Berkeley, CA, USA in 1994. He then joined the Hong Kong University of Science and Technology, Kowloon, Hong Kong, where he is currently a professor in the Department of Electronic and Computer Engineering and the Division of Biomedical Engineering. His research interests are in bio-inspired signal processing and robotics, neuromorphic engineering, computational neuroscience, machine vision, image processing, and machine learning. He has twice served as distinguished lecturer for the IEEE Circuits and Systems Society. He is an associate editor of the *IEEE Transactions on Biomedical Circuits and Systems* and the *Frontiers in Neuromorphic Engineering*. He is a fellow of the IEEE.

Ryad B. Benosman is a full professor in the Université Pierre et Marie Curie, Paris, France. He leads the Neuromorphic Vision and Natural Computation group. His interest focuses mainly on neuromorphic engineering, visual computation and sensing. He is a pioneer of omnidirectional vision systems, complex perception systems, variant scale sensors, and non-central sensors. His current research interests include the understanding of the computation operated by the visual system with the goal to establish the link between computational and biological vision. He is also invested in applying neuromorphic computation to retina prosthetics and he is a cofounder of the startup Pixium Vision.