Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate Coding and Coincidence Processing—Application to Feedforward ConvNets

José Antonio Pérez-Carrasco, Bo Zhao, *Student Member*, *IEEE*, Carmen Serrano, *Member*, *IEEE*, Begoña Acha, *Member*, *IEEE*, Teresa Serrano-Gotarredona, Shouchun Chen, *Member*, *IEEE*, and Bernabé Linares-Barranco, *Fellow*, *IEEE*

Abstract—Event-driven visual sensors have attracted interest from a number of different research communities. They provide visual information in quite a different way from conventional video systems consisting of sequences of still images rendered at a given "frame rate." Event-driven vision sensors take inspiration from biology. Each pixel sends out an event (spike) when it senses something meaningful is happening, without any notion of a frame. A special type of event-driven sensor is the so-called dynamic vision sensor (DVS) where each pixel computes relative changes of light or "temporal contrast." The sensor output consists of a continuous flow of pixel events that represent the moving objects in the scene. Pixel events become available with microsecond delays with respect to "reality." These events can be processed "as they flow" by a cascade of event (convolution) processors. As a result, input and output event flows are practically coincident in time, and objects can be recognized as soon as the sensor provides enough meaningful events. In this paper, we present a methodology for mapping from a properly trained neural network in a conventional frame-driven representation to an event-driven representation. The method is illustrated by studying event-driven convolutional neural networks (ConvNet) trained to recognize rotating human silhouettes or high speed poker card symbols. The event-driven ConvNet is fed with recordings obtained from a real DVS camera. The event-driven ConvNet is simulated with a dedicated event-driven simulator and consists of a number of event-driven processing modules, the characteristics of which are obtained from individually manufactured hardware modules.

Index Terms— Feature extraction, convolutional neural networks, object recognition, spiking neural networks, event-driven neural networks, bioinspired vision, high speed vision

1 Introduction

In 2006, Delbrück et al. [1], [2] presented the first event-driven dynamic vision sensor (DVS) inspired by Kramer's transient detector concept [3]. This was followed and improved by other researchers [4], [5]. The DVS presents a revolutionary concept in vision sensing as it uses an event-driven frameless approach to capture transients in visual scenes.

A DVS contains an array of pixels (i, j) where each pixel senses local light I_{ij} and generates an asynchronous "address event" every time light changes by a given relative amount $C > 1$ (if light increases: when $I_{ij}(t)/I_{ij}(t_o) = C$, or if light decreases: when $I_{ij}(t)/I_{ij}(t_o) = 1/C$). The "address event" consists of the pixel coordinates (x_{ij}, y_{ij}) and sign s_{ij} of the change (increment or decrement). This "flow" of asynchronous events is usually referred to as "address event representation" (AER). Every time a DVS pixel generates such an event, the event parameters (x_{ij}, y_{ij}, s_{ij}) are written on a high speed asynchronous digital bus with nanosecond delays. A DVS pixel typically generates one to four events (spikes) when an edge crosses it. DVS output consists of a continuous flow of events (spikes) in time, each with submicrosecond time resolution, representing the observed moving reality as it changes, without waiting to assemble or scan artificial time-constrained frames (images).

As an illustration, Fig. 1 shows the event flow generated by a DVS when it observes a black 400-Hz rotating disk with a white dot. On the right, events are represented in 3D coordinates (x, y, t). When a pixel senses a dark-to-bright transition, it sends out positive events (dark dots in Fig. 1),

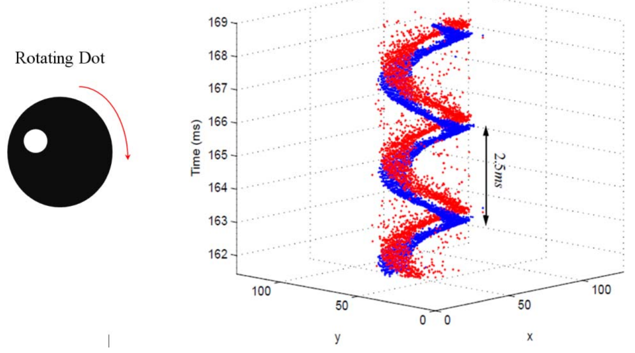


Fig. 1. Example illustration of DVS camera output event flow when observing a black rotating disk with a white dot, rotating at 400 Hz.

and when it senses a bright-to-dark transition, it sends out a negative event (gray dots in Fig. 1). Appendix 1, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2013.71, explains the operation of a typical DVS camera in more detail. The flow of events generated by a DVS can be captured with an event logger board [6], [7], and written on a file with corresponding time stamps. This file contains a list of signed time-stamped events (t, x, y, s).

Recorded time-stamped events can be processed offline to perform filtering, noise removal, shape detection, object recognition, and other operations. However, it is more desirable to develop event-driven processing hardware to process events as they are generated by the DVS, without time stamping them, and to operate in true real time. For example, some event-driven convolution processors have recently been reported for performing large programmable kernel 2D convolutions on event flows [8], [9], [10]. Appendix 2, available in the online supplemental material, briefly explains the operation of a typical AER programmable kernel convolution chip. One very interesting property of this event-driven processing is what we call here "*pseudosimultaneity*" or "*coincidence*" between input and output event flows. This concept is illustrated with the help of Fig. 2. A vision sensor is observing a flashing symbol that lasts for 1 ms. The sensor then sends its output to a five-layer convolutional neural network for object recognition, as shown in Fig. 2a. In the case of conventional frame-driven sensing and processing, the sequence of processing results would be as depicted in Fig. 2b. Assuming sensor and each processing stage respond in 1 ms, the sensor output image would be available during the next millisecond after the flash. Then, each sequential stage would provide its output 1 ms after receiving its input. Therefore, recognition (last stage output) becomes available 6 ms after the symbol flashed. Fig. 2c shows the equivalent when sensing and processing with event-driven hardware. Pixels in the sensor create and send out events as soon as they sense a light change, with microseconds delay [1], [2], [4], [5]. This way, the sensor output events at x_o are, in practice, simultaneous to the flashing symbol in reality. The first event-driven stage processes events as they flow in, with submicrosecond delays [8], [9], [10], [11]. As soon as sufficient events are received representing a given feature, output events will be available. Thus, the output feature

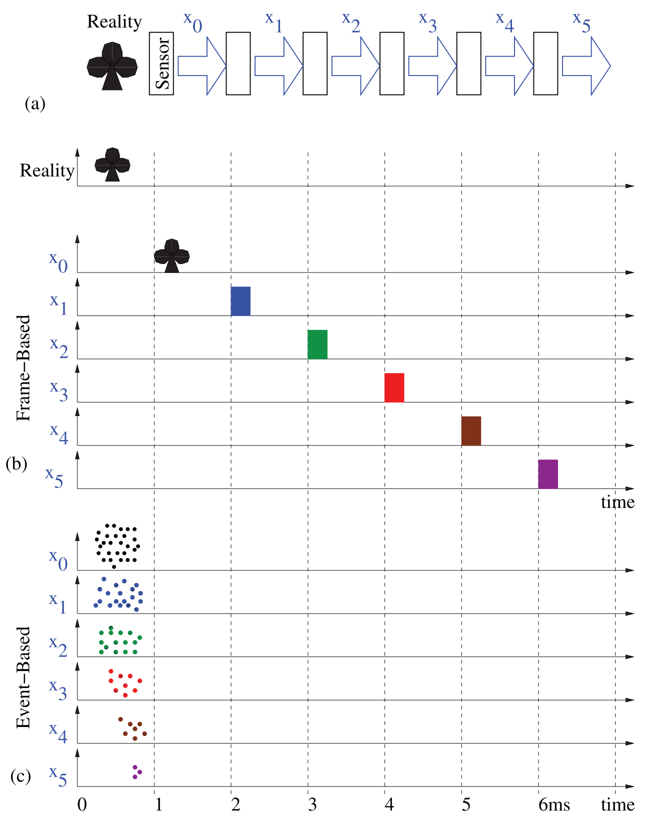


Fig. 2. Illustration of pseudosimultaneity or coincidence property in a multilayer event-driven processing system. (a) Vision system composed of vision sensor and five sequential processing stages, like in a ConvNet. (b) Timing in a frame-constraint system with 1-ms frame time for sensing and per stage processing. (b) Timing in an event-driven system with microsecond delays for sensor and processor events.

event flow at x_1 is, in practice, coincident with the event flow at x_o. The same happens for the next stages. Therefore, recognition at x_5 becomes available during the first millisecond, as soon as the sensor provides sufficient events for correct recognition.

This *pseudosimultaneity* or *coincidence* property becomes very attractive for event-driven processing systems comprising a large number of cascaded event-driven processors with or without feedback, as the overall output can be available as soon as sufficient meaningful input events are provided. This contrasts strongly with state-of-the-art frame-driven vision sensing and processing, where images are first detected by a camera and then transferred to an image processor.

In this paper, we focus on vision systems comprising an event-driven sensor and a large number of event-driven processing modules used to perform object recognition tasks.[1] To do so, we will concentrate on a particular type of bioinspired vision processing structures called convolutional neural networks (ConvNets) [12]; reported ConvNets operate based on frame-driven principles and are trained by presenting them with a database of training static images (frames). On the other hand, training of event-driven processing modules is still an open research problem. Some

1. As discussed in Appendix 2, available in the online supplemental material, with present day technology, it is feasible to develop compact hardware with thousands of event-driven convolution modules [28].

preliminary and highly promising work on this can be found in the literature [19], [20]. However, its application to large-scale systems is presently not practical. Therefore, in this paper, we present an intermediate solution. First, we build a database of training images (frames) by collecting events from a DVS camera during fixed time intervals. Second, we train a frame-driven ConvNet with this database to perform object recognition. Third, we map the learned parameters of the frame-driven ConvNet to an event-driven ConvNet, and finally, we fine-tune some extra available timing-related parameters of the event-driven ConvNet to optimize recognition. To do this process, we provide a methodology for mapping the properly trained frame-driven ConvNet into its corresponding event-driven version. We will then illustrate this with two example ConvNet exercises: one for detecting the angle of rotated DVS recordings of walking human silhouettes, and the other for recognizing the symbols of poker cards when browsing the card deck in about 1 second in front of a DVS.

The paper is structured as follows: The next section discusses timing differences between vision in frame-driven and event-driven representations. Section 3 presents the mapping method from a frame-driven system neuron to an event-driven system neuron. Sections 4 and 5 present two example ConvNet systems that use DVS recordings from real DVS retina chips. In Section 4, the example targets a problem where the time constants of the observed world are similar to those we humans are used to, while the experiment in Section 5 illustrates the situation for higher speed observed realities where DVS performance is pushed to its limits. Finally, Sections 6 and 7 present some discussions and the conclusions.

2 Timing in Frame-Driven versus Event-Driven Vision Representation

In a frame-driven visual processing system "*reality*" is sensed as binned into time compartments of duration T_{frame}. The implicit assumption is that the time constant τ_{reality} associated with the changes in "*reality*" is larger than T_{frame} or, at most, similar. If τ_{reality} is much larger than T_{frame} (reality moves slowly), then many subsequent video frames would be quite similar and redundant. An image capturing and processing system working on a frame by frame basis would repeat complex image processing and recognition algorithms over a similar input, wasting computing resources. If τ_{reality} is much smaller than T_{frame} (reality moves very fast), then subsequent video frames would be considerably different, making it difficult or impossible to track objects (for example, many flies in a box). Optimally, one would desire to adjust T_{frame} to be close to τ_{reality} so that subsequent frames are different enough to justify the computing resources employed, but still similar enough to be able to track changes.

In an event-driven vision sensing and processing system, frames need not be used. For example, in event-driven temporal contrast retina sensors (DVS), pixels generate output events representing "*moving reality*" with time constants that adapt naturally to τ_{reality}. In the particular case of feedforward multilayer ConvNets, subsequent layers extract visual features that are simple and short range in the first layers and progressively become more and more complex and longer range in subsequent layers until specific full-scale objects are recognized. Typically, first layers extract edges and orientations at different angles and scales, using short range but dense projection fields (receptive fields). Subsequent layers group these simple features progressively into gradually more sophisticated shapes and figures, using longer range but sparser projection fields. Here, we assume that the processing time constants associated with the first feature extraction layers are faster than those associated with later layers. This way, early feature extraction layers would be short range both in space and time, while later feature grouping layers would be longer range also in both space and time. Note that this makes a lot of sense, since simple features (such as short edges) need to be sensed instantly, while for recognizing a complex shape (like a human silhouette) it would be more efficient to collect simple features during a longer time to be more confident. For example, if we observe a walking human silhouette, at some instants we may not see a leg or an arm, but if we see them at other instants we know they are there. Consequently, in a frame-free event-driven sensing and processing system, we have the extra feature of adapting the time constant of each processing layer independently. This provides extra freedom degrees to optimize overall recognition, which is not directly available in frame-driven recognition systems.

At present, however, frame-driven vision machine learning algorithms are much more developed than their event-driven counterparts. For example, over the last few decades, powerful and highly efficient training algorithms have been developed and applied for frame-driven ConvNets, making them practical and competitive for a variety of real-world applications [12], [13], [14], [15], [16], [17], [18]. Some researchers are presently exploring the possibility of training event-driven systems with promising results [19], [20], [21]. But this field is still under development.

In the next section, we describe a method for mapping the parameters of a properly trained frame-driven neuron into the equivalent event-driven frame-free parameters. We then illustrate this by applying it in ConvNet visual recognition systems that use real recordings from a frame-free event-driven DVS retina chip.

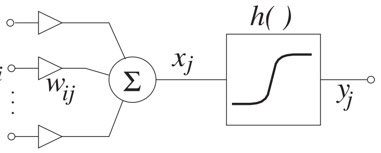


Fig. 3. Conventional individual neuron used in frame-driven systems by freezing time during each frame and resetting its state for each frame.

3 Generic Mapping Methodology

3.1 Frame-Driven Individual Neuron

Fig. 3 shows the computational diagram of a typical neuron in a frame-driven representation system. Input signals y_i come from the ith neuron of the receptive field RF_j of neuron j, weighted by synaptic weights w_{ij}. Input signals y_i belong to range $[0, A_i]$ or $[-A_i, A_i]$. Let us call $y = \hat{y}A$ so that

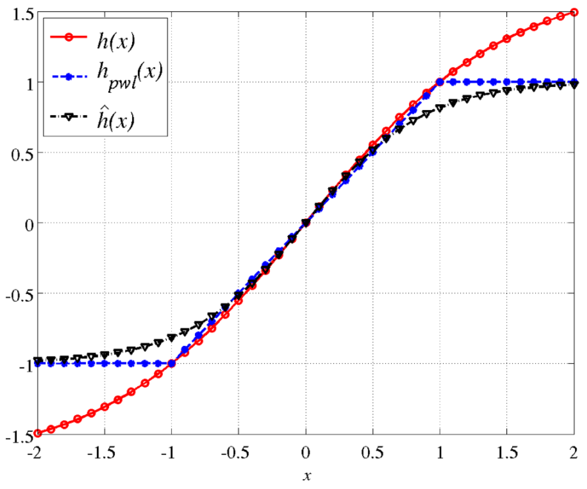


Fig. 4. Comparison between the three nonlinear functions $h(x)$, $h_{pwl}(x)$, and $\hat{h}(x)$.

$\hat{y}$ is normalized to unity. The state x_j of neuron j is reset for each frame, and computed for its receptive field for the present frame as

$$\begin{aligned} x_j &= \sum_{i \in RF_j} y_i w_{ij} = \sum_{i \in RF_j} A_i \hat{y}_i w_{ij} = A_{RF_j} \hat{x}_j, \\ \hat{x}_j &= \sum_{i \in RF_j} \hat{y}_i w_{ij}, \end{aligned} \tag{1}$$

where we have assumed that all A_i coefficients are the same for all neurons i of the RF_j receptive field $A_i = A_{RF_j}$. After this, the neuron state goes through a sigmoidal function $h(\cdot)$, which we may define as[2] [22]

$$\begin{aligned} y_j &= h(x_j) = A_j \tanh\,(S_j x_j) = A_j \tanh\,(S_j A_{RF_j} \hat{x}_j), \\ \hat{y}_j &= \tanh\,(S_j A_{RF_j} \hat{x}_j). \end{aligned} \tag{2}$$

We can describe this using only normalized variables as

$$\begin{aligned} \hat{y}_j &= \hat{h}(\hat{x}_j), \\ \hat{x}_j &= \sum_{i \in RF_j} \hat{y}_i w_{ij}, \end{aligned} \tag{3}$$

with $\hat{h}(z) = \tanh(S_j A_{RF_j} z) = (1/A_j) h(z/A_{RF_j})$. A piecewise-linear approximation of $\hat{h}(\cdot)$ can be defined as

$$h_{pwl}(x) = \begin{cases} x & \text{if } |x| \leq 1 \\ x/|x| & \text{if } x \geq 1. \end{cases} \tag{4}$$

Fig. 4 shows the three nonlinear functions $h(x)$, $h_{pwl}(x)$, and $\hat{h}(x)$ for $S = 2/3$ and $A_j = A_{RF_j} = 1.7159$.

3.2 Event-Driven Individual Neuron

Fig. 5 shows a schematic computational diagram of an event-driven (spiking signal) neuron. In this case, time plays a crucial role as opposed to the previous case where time is frozen during all computations corresponding to a frame. Now, the neural state x'_j evolves continuously with time. Fig. 5 represents state x'_j as being held in a box, while the elements capable of altering it have been drawn with arrows pointing toward this box. These elements are: 1) synaptic connections, 2) leak, and 3) a "reset and refractory" (*R&R*) element.

Presynaptic neurons belonging to the receptive field send spikes in time. In general, spikes carry a positive or negative sign, and synaptic weights also have a positive or negative sign. In certain implementations (such as biology), positive and negative events (spikes) are separated into separate paths. Our analyses are not affected by how this is implemented physically. Each presynaptic spike will contribute to a certain increment or decrement $\Delta x'$ in the neuron state x'_j proportional to the corresponding synaptic weight $|w'_{ij}|$. The neuron state x'_j will accumulate all these contributions over time, and at a given instant may have a positive or negative accumulated value.

Fig. 6 shows an example of neural state evolution and spike production. Let us define a characteristic time T_{C_j} for neuron j. A neuron can be considered a specific feature detector for the collection of spatiotemporal input spikes it receives. For example, neurons in cortex layer V1 spike when they detect sequences of spikes from the retina representing edges at specific scales, orientations, and positions, within a characteristic time interval. A neuron at a higher layer may be specialized in detecting specific shapes, like an eye, nose, and so on. Such a neuron would generate spikes when the collection of input spikes from prior neurons represents a collection of edges and shapes that when put together during a characteristic time interval resemble an eye, nose, and so on. In Fig. 6, we have represented T_{C_j} as a characteristic time during which neuron j receives a meaningful collection of spikes (representing the specific feature of neuron j) that produce a systematic increase in its state. Every time the state x'_j reaches one of the thresholds $\pm x_{thj}$, the *R&R* element will reset the state to its resting level x_{rest} while also guaranteeing a minimum separation between consecutive spikes T_{R_j}, called the "refractory time" of this neuron. This refractory effect is equivalent to the saturation function $h(\cdot)$ in the frame-driven system as it limits the maximum output spike event rate.

If all neurons i of the receptive field of neuron j have the same characteristic time T_{C_i} and/or refractory time T_{R_i}, we can define the "characteristic time gain" of neuron j as

$$g_{\tau j} = T_{C_j} / T_{C_i}, \tag{5}$$

and the "refractory time gain" of neuron j as

$$g_{Rj} = T_{R_j} / T_{R_i}. \tag{6}$$

We will use these definitions later.

2. LeCun [22] suggested setting $A = 1.7159$ and $S = 2/3$ to optimize learning speed and convergence in ConvNets.

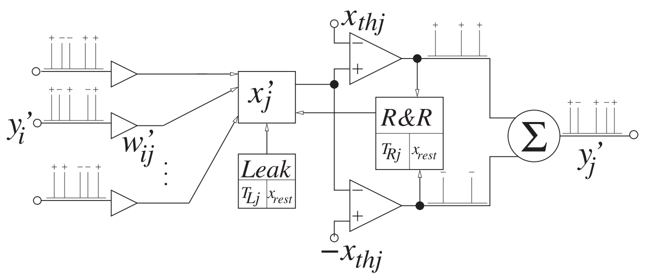


Fig. 5. Computational block diagram of event-driven neuron.

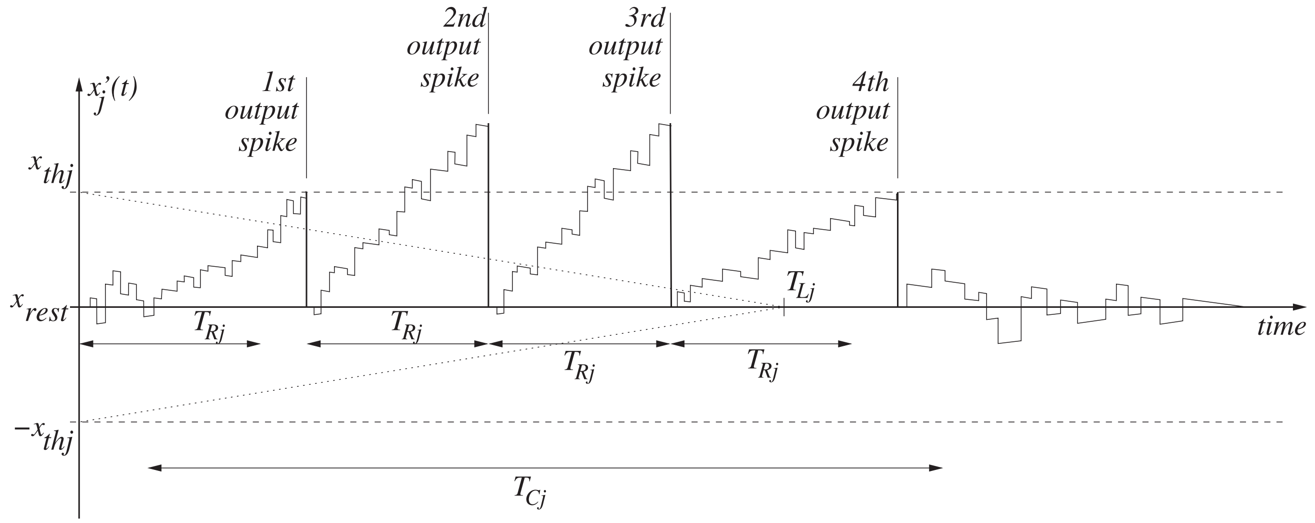


Fig. 6. Illustration of a typical state evolution and spike production sequence for a spiking neuron with leak and refractory period.

Neurons will not accumulate all historic incoming spikes contributions (similarly, in the frame-driven case, neurons ignore information from previous frames). Since a neuron is interested in grouping lower level features from previous neurons during a characteristic time T_{C_j}, its state x'_j is subject to a continuous leak that will drive its value toward x_{rest} with a characteristic leak time constant. Fig. 6 shows a linear leak for the neuron state, with a leak rate of value $LR_j = |x_{th_j}/T_{L_j}|$.

3.3 Encoding in Frame-Free Event-Driven Systems—Low-Rate Rate Coding or Coincidence Processing

In traditional frame-driven neural computing systems, neuron states and neuron output values are usually represented with floating-point precision. In some specialized accelerated hardware implementations, 8-bit signed integer representation is used [23]. Still, this representation presents a high dynamic range since the ratio between the full range and the smallest step is $2^8 = 256$. Note that in a recognition system, the output of a neuron is not required to present such a high dynamic range because it only has to signal whether a feature is present or at the most provide a relative confidence that could be provided with coarse steps. For example, in a face detection application, we would not expect it to be critical whether the neurons detecting the nose can use just five values $[0, 0.25, 0.50, 0.75, 1.0]$ to give their confidence, or can use 256 steps in the range $[0, 1]$. A higher dynamic range might be necessary to represent the visual input. Commercial video, photography, and computer screens normally use 8-bit to represent luminance. However, we will assume that our event-driven visual sensors include a preprocessing step (such as spatial or temporal contrast) that significantly reduces the dynamic range of the signals provided by their pixels. For example, a pixel in the temporal contrast DVS retina we have used normally provides between one to four spikes when an edge crosses it.

In the following mathematical developments for mapping from the frame-driven domain to the event-driven domain, we will consider that an intensity value in the former is mapped to a spike rate in the latter. Obviously, rate coding is highly inefficient for high dynamic ranges such as 8 bit, because a neuron would need to transmit 256 spikes to represent a maximally meaningful signal. Although the following mathematical developments have no restrictions in terms of dynamic range, we will always keep in mind that we will in practice apply it to low dynamic range signals. We call this "low-rate rate coding," and the maximum number of spikes a neuron will transmit during its characteristic time constant will be kept relatively low (for example, just a few spikes, or even as low as one single spike). This maximum number of spikes is T_{C_j}/T_{R_j}. Time T_{R_j} is the minimum interspike time needed to signal the presence of a feature, while T_{C_j} is the characteristic time during which this feature might be present during a transient. Thus, let us call "persistency" p_j of neuron j the maximum number of spikes that can be generated by a transient feature during time T_{C_j}

$$p_j = T_{C_j}/T_{R_j}. \tag{7}$$

Using all the above concepts and definitions, let us now proceed to mathematically analyze the event-driven neuron and propose a mapping formulation between frame-driven and event-driven neuron parameters.

3.4 Mathematical Analysis of Event-Driven Neurons

With reference to Fig. 6, let us consider the situation where neuron j becomes active as it receives a collection of properly correlated spatiotemporal input spikes during a time T_{C_j}. These represent the feature to which neuron j is sensitive. In this case, the collection of spikes will produce a systematic increase in neuron j's activity x'_j during time T_{C_j}, resulting in the generation of some output spikes, as illustrated in Fig. 6. If the output spikes are produced with interspike intervals larger than the refractory period T_{R_j}, then the number of spikes n_j produced by neuron j during time T_{C_j} satisfies[3]

3. Here, we are assuming a positive increase in the state ($x'_j > x_{\text{rest}}$), reaching the positive threshold and producing positive output events. The analysis is equivalent for the generation of negative events.

TABLE 1
Summary of Event-Driven Neuron Parameters

x_{thj}	threshold
T_{Cj}	characteristic time
T_{Rj}	refractory time
LR_j	leak rate
$p_j = T_{Cj}/T_{Rj}$	persistency
$g_{\tau j} = T_{Cj}/T_{Ci}$	characteristic time gain
$g_{Rj} = T_{Rj}/T_{Ri} = r_{ij}$	refractory time gain
$\beta_j = T_{Rj}/T_{Lj}$	refractory-leak ratio

$$\frac{n_j}{T_{Cj}} = \frac{\left(\sum_{i\in RF_j} n_i w'_{ij}\right) - \Delta x_{Lj}}{x_{thj}T_{Cj}}, \tag{8}$$

where Δx_{L_j} is the loss of neural activity x'_j due to a leak. We may safely assume that during a systematic neural activity build up that produces output events, a leak does not drive the activity down to the resting level x_{rest}, nor does it produce a change of its sign. Under this assumption,

$$LR_j = \frac{|\Delta x_{L_j}|}{T_{C_j}} = \frac{x_{th_j}}{T_{L_j}}. \tag{9}$$

If the systematic activity build up is sufficiently fast, then the neuron activates its refractory mode and will not allow interspike intervals shorter than T_{R_j}, or equivalently

$$\frac{n_j}{T_{C_j}} \leq \frac{1}{T_{R_j}} \tag{10}$$

as is illustrated in Fig. 6 for the second and third spikes produced by neuron j during time T_{C_j}. To take this into account, the right-hand side of (8) needs to saturate to $1/T_{R_j}$. This can be expressed as

$$\frac{n_j}{T_{C_j}} = \frac{1}{T_{R_j}} h_{pwl}\left(\frac{\left(\sum_{i\in RF_j} n_i w'_{ij}\right) - \Delta x_{L_j}}{x_{th_j}} \frac{T_{R_j}}{T_{C_j}}\right) \leq \frac{1}{T_{R_j}}, \tag{11}$$

where $h_{pwl}(\cdot)$ saturates to "1" and is as defined in (4). Using (7) and (9), (11) becomes

$$\frac{n_j}{p_j} = h_{pwl}\left(\left(\sum_{i\in RF_j} \frac{n_i}{p_j}\frac{w'_{ij}}{x_{th_j}}\right) - \beta_j\right), \tag{12}$$

where $\beta_j = T_{R_j}/T_{L_j}$. Noting that β_j will usually tend to be much smaller than unity and that $n_j/p_j \in [-1, 1]$, we can establish a parallelism between bottom (2) and (12) by means of the following mapping:

$$\begin{aligned} \hat{y}_j &\leftrightarrow \frac{n_j}{p_j}, \quad \hat{y}_i \leftrightarrow \frac{n_i}{p_i}, \\ w_{ij} &\leftrightarrow \frac{w'_{ij}}{x_{th_j}}\frac{p_i}{p_j} = \frac{w'_{ij}}{x_{th_j}}\frac{g_{R_j}}{g_{\tau_j}}, \\ \hat{h}(\cdot) &\leftrightarrow h_{pwl}(\cdot). \end{aligned} \tag{13}$$

Note that the kernel weights w'_{ij} used for the event-driven realization are simply scaled versions of those trained in the frame-based version w_{ij}. Table 1 summarizes the different event-driven neuron parameters discussed. As we will see in the rest of the paper, when applying this mapping to ConvNets we will use the same mapping for all neurons in the same ConvNet layer.

It is interesting to highlight that in a frame-driven system, neuron states $\hat{x}_j$ can be interpreted as showing how much they have changed during the frame time T_{frame} after being reset, and this frame time can in turn be interpreted as the "characteristic time" T_C of all neurons in the system. When mapping from a frame-driven description to an event-driven one, all neurons could therefore be made to have identical timing characteristics. However, as we will see later on, neurons in different ConvNet layers will be allowed to have different timing characteristics to optimize recognition performance and speed.

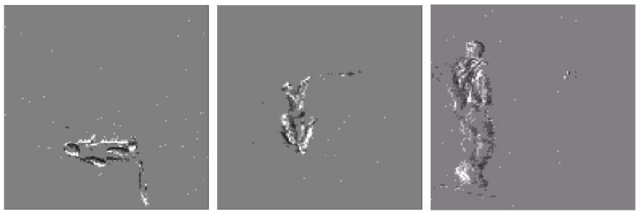


Fig. 7. Example snapshot images obtained by histogramming events during 80 ms and rotating the (x, y) addresses.

4 Event-Driven ConvNet for Human Silhouette Orientation Recognition

As an illustrative example of scenes moving at speeds we humans are used to, we trained a frame-driven version of a ConvNet to detect the orientation of individual human walking silhouettes. We used a 128×128 pixel DVS camera [5] to record event sequences when observing individual people walking. Fig. 7 shows some (x, y) rotated sample images obtained by collecting DVS recorded events during[4] 80 ms. White pixels represent positive events (light changed from dark to bright during these 80 ms), while black pixels represent negative events (light changed from bright to dark). One person walking generates about 10-20 keps (kilo events per second) with this DVS camera. From these recordings, we generated a set of images by collecting events during frame times of 30 ms. From these reconstructed images, we randomly assigned 80 percent for training and 20 percent for testing learning performance. Each 128×128 pixel reconstructed image was downsampled to 32×32 and rotated 0, 90, 180, or 270 degrees.

The training set images were used to train the frame-driven six layer feedforward ConvNet shown in Fig. 8. Table 2 summarizes the number of feature maps (FMs) per layer, FM size, kernels size, total number of kernels per layer, total weights per layer, and how many weights are trainable. The first layer C1 performs Gabor filtering at three orientations and two scales, and its weights are not

4. Recordings were made by connecting the DVS camera via USB to a laptop running jAER [25]. jAER is an open software for managing AER chips and boards, recording events, playing them back, and performing a variety of processing operations on them. Appendix 3, available in the online supplemental material, gives a brief overview of jAER.

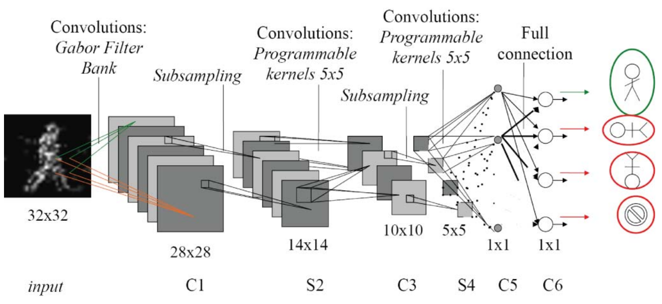

Convolutions: Gabor Filter Bank
Subsampling
Convolutions: Programmable kernels 5x5
Convolutions: Programmable kernels 5x5
Subsampling
Full connection
32x32
28x28
14x14
10x10
5x5
1x1
1x1
input
C1
S2
C3
S4
C5
C6

Fig. 8. ConvNet structure for human silhouette orientation detection.

TABLE 2
ConvNet Structure

	C1	S2	C3	S4	C5	C6
Feature Maps (FM)	6	6	4	4	8	4
FM size	28x28	14x14	10x10	5x5	1x1	1x1
kernels size	10x10	-	5x5	-	5x5	1x1
kernels	6	-	24	-	32	32
weights	600	-	600	-	800	32
trainable weights	0	-	600	-	800	32

trained. S layers perform subsampling and subsequent C layers perform feature extraction and grouping. The last layer (C6) is not a feature extraction layer, but a feature grouping layer. It performs a simple linear combination of the outputs of the previous layer. The top three neurons in layer C6 recognize a human silhouette rotated $0, +/-90$, or 180 degrees. The bottom neuron (noise) is activated when the system does not recognize a human silhouette. Each output neuron fires both positive and negative events, depending on whether it is certain the desired pattern is present or it is certain it is not present.

The weights from the frame-driven system were then mapped to an event-driven version by using the transformations in (13). Note that in a feedforward ConvNet, all neurons in the same layer operate with identical spatial scales (kernel sizes and pixel space sizes). Here, we will enforce that neurons in the same layer operate with identical temporal scales too, so that each layer extracts spatiotemporal features of the same scales. In Table 1, we would therefore replace index j with the layer number index n, and index i with the previous layer index $n-1$. We also chose $w_{ij} = w'_{ij}$ in (13), which enforces $g_{R_n} = x_{th_n} g_{\tau_n}$.

However, (5)-(7) and (13) offer a high degree of freedom to map parameters. We first followed the heuristic rationale outlined below, but afterward we ran simulated annealing optimization routines to adjust the different parameters for optimum performance.

The temporal patterns generated by the 128×128 DVS camera when observing walking humans are such that a minimum time of about 10-20 ms (about 100-600 events) is needed to reconstruct a human-like silhouette.[5] We therefore set the "refractory time" of the last refractory layer (C5) as $T_{R_5} \approx 10$ ms. On the other hand, the persistency of a moving silhouette is on the order of 100 ms (collecting events for over 100 ms fuzzyfies the silhouette). Thus, $T_{C_5} \approx 100$ ms. For layer C1, short-range edges can be observed with events separated about 0.1 ms in time. We therefore set $T_{R_1} \approx 0.1$ ms. For layer C3, we chose an intermediate value $T_{R_3} \approx 0.5$ ms. For the thresholds, we picked a value approximately equal to twice the maximum kernel weight projecting to each layer. For the leak rates, we picked an approximate ratio of 2:1 between consecutive layers, so that the last layer C6 has a leak rate of $1\ \mathrm{s}^{-1}$. With these criteria, and considering that $g_{R_n} = x_{th_n} g_{\tau_n}$, the resulting list of parameters describing the event-driven system is shown in Table 3.

Despite this heuristic method of obtaining a set of timing parameters for the six layer event-driven ConvNet, we also used optimization routines to optimize these parameters for best recognition rate, as mentioned in the next section and described in detail in Appendix 5, available in the online supplemental material.

5. A retina with higher spatial resolution (256×256 or 512×512) multiplies the number of events produced (by 4 and by 16, respectively) for the same stimulus, but the events would still be produced during the same 10-20-ms time interval. Therefore, we conjecture that increasing spatial resolution reduces recognition time because the 100-600 events for first recognition would be available earlier.

4.1 Results

For testing the event-driven system, we used new recordings from the 128×128 pixel DVS camera observing people. These recordings were downsampled to the 32×32 input space.

To run the simulations on these recordings, we used the address-event-representation event-driven simulator AERST (AER Simulation Tool) [24]. This simulator is briefly described in Appendix 4, available in the online supplemental material. It uses AER processing blocks, each with one or more AER inputs and one or more AER outputs. AER outputs and inputs are connected through AER point-to-point links. Therefore, the simulator can describe any AER event-driven system through a netlist of AER blocks with point-to-point AER links. A list of DVS recordings is provided as stimulus. The simulator looks at all AER links and processes the earliest unprocessed event. When an AER block processes an input event, it may generate a new output event. If it does, it will write a new unprocessed event on its output AER link. The simulator continues until there are no unprocessed events left. At this point, each AER link in the netlist contains a list of time-stamped events. Each list represents the visual flow of spatiotemporal features represented by that link. The resulting visual event flow at each link can be seen with the jAER viewer.

TABLE 3
Parameters Adjusted by Heuristic Rationale

layer	T_{Rn}	x_{thn}	*leak rate*	T_{Ln}	T_{Cn}	p_n	g_{Rn}	$g_{\tau n}$	β
C1	$0.1ms$	0.6	$10s^{-1}$	$60ms$	$5ms$	40	-	-	0.0017
C3	$0.5ms$	1	$5s^{-1}$	$200ms$	$25ms$	40	5	5	0.005
C5	$10ms$	5	$2s^{-1}$	$2.5s$	$100ms$	10	20	4	0.004
C6	-	1.5	$1s^{-1}$	$1.5s$	-	-	-	-	-

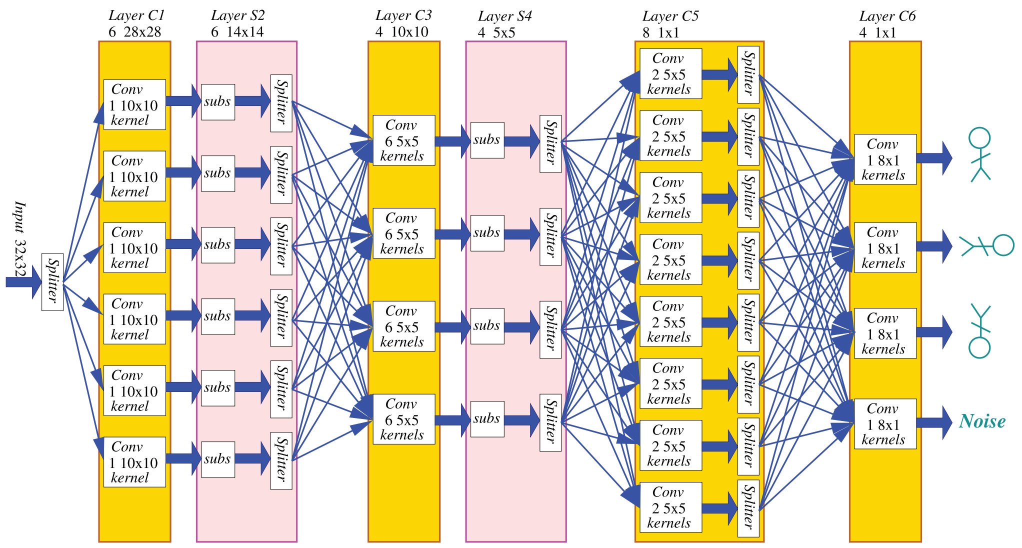


Fig. 9. Schematic block diagram used in simulator AERST.

Fig. 9 shows the netlist block diagram of the event-driven ConvNet system simulated with AERST. It contains 19 splitter modules, 20 AER convolution modules, and 10 subsampling modules. In AERST, the user can describe modules by defining the operations to be performed for each incoming event and include nonideal effects such as characteristic delays, noise and jitter, limited precision, and so on. We described the modules using the performance characteristics of already manufactured AER hardware modules, specifically, available splitter and mapper modules implemented on FPGA boards [6], [9], and dedicated AER convolution chips with programmable kernels [8], [10], [11]. A splitter module replicates each input event received at each of its output ports with a delay on the order of 20-100 ns. Subsampling modules are implemented using mappers. They transform event coordinates. In this particular case, the mappers are programmed to replace each input event coordinate (x, y) with $(\lfloor x/2 \rfloor, \lfloor y/2 \rfloor)$, where operand $\lfloor z \rfloor$ is "round to the lower integer." This way, all events coming from a 2×2 square of pixels are remapped to a single pixel. Convolution modules describe AER convolution chip operations with programmable kernels. Convolutions are computed and updated event by event, as described in Appendix 2, available in the online supplemental material.

Using the parameters in Table 3, we tested the performance of the event-driven ConvNet in Fig. 9 when fed with event streams of DVS captured walking human silhouettes rotated 0, 90, 180, and 270 degrees. Each input stream segment consists of 2k events, and their (x, y) coordinates are consecutively rotated 0, 90 or 270, and 180 degrees. Fig. 10 shows input events as small black dots. Output events are marked either with circles ("*upright*" or 0 degree), crosses ("*horizontal,*" or 90 and 270 degrees), or stars ("*upside down*" or 180 degree). Fig. 10b shows a zoom out for the first transition from horizontal to upside down, where the recognition delay is 17 ms. Average recognition delay was 41 ms, and the fastest was 8.41 ms. The overall recognition success rate *SR* for this stream was 96.5 percent, computed as the average of the success rate per category SR_i. For each category i, its success rate is computed as

$$SR_i = \frac{1}{2}\left(\frac{p_i^+}{p_i^+ + p_i^-} + \frac{p_{j\neq i}^-}{p_{j\neq i}^+ + p_{j\neq i}^-}\right), \tag{14}$$

where p_i^+ (p_i^-) is the number of positive (negative) output events for category i when input stimulus corresponds to category i, and $p_{j\neq i}^+$ ($p_{j\neq i}^-$) are the positive (negative) output events for the other categories. In a perfect recognition situation, $p_i^- = 0$ and $p_{j\neq i}^+ = 0$.

The parameters in Table 3 were assigned by intuition. Appendix 5, available in the online supplemental material, describes the results obtained when using a simulated annealing optimization procedure to optimize these parameters. The resulting optimized parameters are not too different from the ones obtained by intuition, and the recognition rate varied within the range 97.28 to 99.61 percent.

To compare recognition performance with that of a frame-driven ConvNet realization, we used the same sequence of events and built sequences of frames using different frame reconstruction times. Each frame was fed to the frame-driven ConvNet. Table 4 shows, for each frame

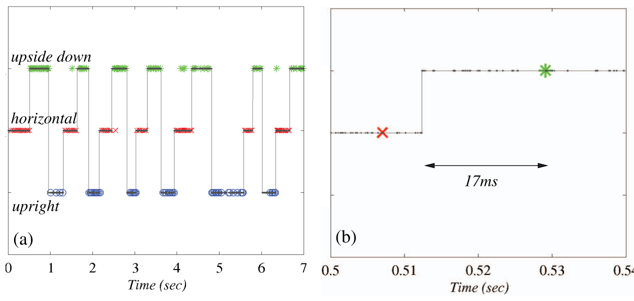


Fig. 10. Recognition performance of the event-driven ConvNet in Fig. 8. Small black dots correspond to input events, circles are output events for "upright" orientation (0 degrees), crosses for "horizontal" orientation (90, 270 degrees), and stars for "upside down" (180 degrees). (a) 7 sec recording of 20 consecutive orientations, (b) zoom out of 40 ms showing a recognition delay of 17 ms.

TABLE 4
Performance of Frame-Driven Realization
of the Human Silhouette Orientation Detection ConvNet

Frame Time	Total non-empty Frames	Success Rate
125ms	45	95.6%
100ms	63	96.8%
75ms	84	97.6%
50ms	133	97.7%
30ms	225	96.4%
20ms	339	96.2%
10ms	674	93.5%

TABLE 5
Performance of Frame-Driven Realization
of the Poker Card Symbol Recognition ConvNet

Frame Time	Total non-empty Frames	Success Rate
10ms	30	63.3%
7ms	45	77.8%
5ms	54	85.2%
4ms	62	91.9%
3ms	75	94.7%
2ms	84	95.2%
1ms	95	92.6%

reconstruction time, the total number of nonempty frames (some frames were empty because the sensor was silent during these times) and the percent of correctly classified frames. As can be seen, the success rate changes with frame reconstruction time and seems to have an optimum in the range 50-75 ms frame time.

5 Event-Driven ConvNet for Poker Card Symbol Recognition

In this section, we illustrate the method with a second example, more oriented toward high speed sensing and recognition. A video illustrating this experiment is available in the online supplemental material. Fig. 11a shows an individual browsing a poker card deck. A card deck can be fully browsed in less than 1 second. When recording such a scene with a DVS and playing the event flow back with jAER, one can freely adjust the frame reconstruction time and frame play back speed to observe the scene at very low speed. Fig. 11b illustrates a reconstructed frame when setting frame time to 5 ms. The DVS had 128×128 pixels. A poker symbol fits well into a 32×32 pixel patch. We made several high speed browsing recordings, built frames of 2 ms time, and cropped many versions of poker symbols of size 32×32 pixels. We selected a set of the best looking frames to build a database of training and test patterns. The topological structure of the frame-driven ConvNet used for recognizing card symbols was identical to the one described in the previous section.

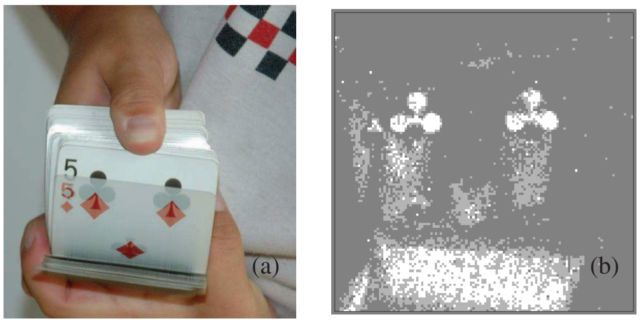


Fig. 11. Fast browsing of a poker card deck. (a) Picture taken with a frame-driven camera. (b) jAER image obtained by collecting events during 5 ms.

The trained frame-driven ConvNet was mapped to an event-driven version by using the same sets of learned kernel weights and adjusting the timing parameters to the higher speed situation. To provide a proper 32×32 pixel input scene, we used an event-driven clustering-tracking algorithm [26] to track card symbols from the original 128×128 DVS recordings from the instant they appeared until they disappeared. Such time intervals typically ranged from about 10-30 ms per symbol and the 32×32 crop could contain on the order of 3k to 6k events. We sequenced several of these tracking crops containing all symbols and used the sequences as inputs to the event-driven ConvNet. We then tested the event-driven ConvNet with the event-driven AER simulator used in Section 4 and used the same Matlab simulated annealing routines to optimize timing parameters. Table 2 in Appendix 5, available in the online supplemental material, shows the optimized parameters and performance results of several optimizations after running simulated annealing. The recognition success rate, measured by (14), varied between 90.1 and 91.6 percent.

To compare the recognition performance with that of a frame-driven ConvNet realization, we used the same input event recordings to generate frames with different frame times, from 1 to 10 ms. Then, we exposed the originally trained frame-driven ConvNet to these new frames and obtained the recognition rates shown in Table 5. As can be seen, there is an optimum frame time between 2 and 3 ms.

Fig. 12 shows an example situation of the output recognition events of the event-driven ConvNet while a sequence of 20 symbol sweeps was presented. The continuous line indicates which symbol was being swept at the input, and the output events are represented by different markers for each category: circles for “club” symbol, crosses for “diamond” symbol, inverted triangles for “heart” symbols, and noninverted triangles for “spade” symbols. Fig. 13 shows the details of the 14 ms sequence of the fourth

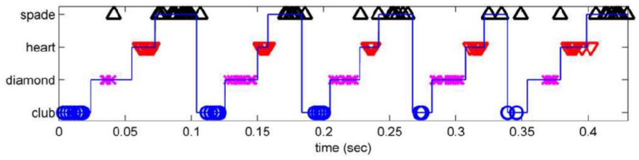


Fig. 12. Recognition performance of the poker card symbol recognition ConvNet.

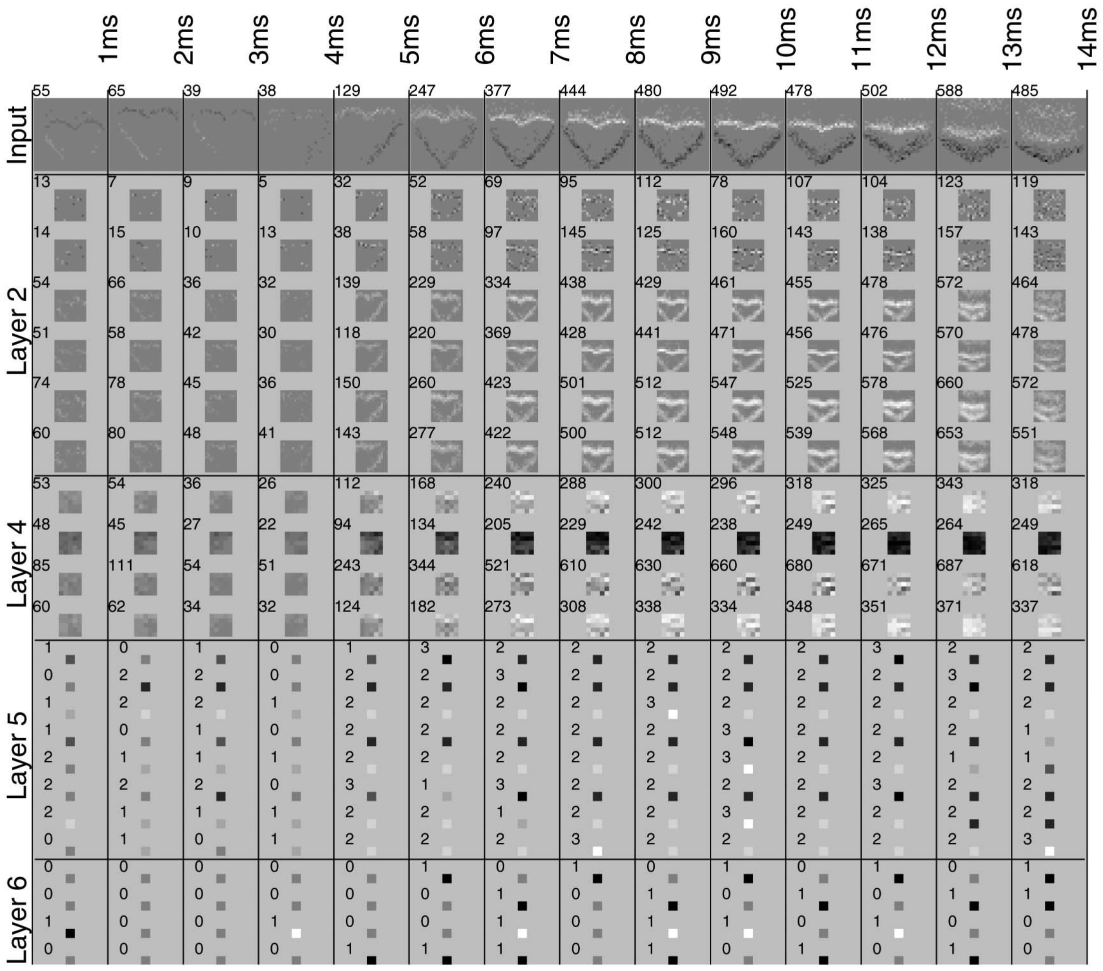


Fig. 13. Image reconstruction for event flows at different ConvNet stages during a 14 ms heart symbol card browsing. Images in this figure are built by collecting events during 1 ms. Event flows are continuous throughout the full 14 ms shown, but are artificially grouped in 1 ms bins. Time advances from left to right in steps of 1 ms. Images in the same column use events collected during the same ms. Horizontal rows correspond to one single feature map output. The top row is a 32×32 pixel crop of the DVS output tracking a 14 ms heart symbol sweep through the screen. The next six rows correspond to Layer S2 subsampled 14×14 pixel feature maps. The next four rows correspond to Layer S4 subsampled 5×5 pixel feature maps. The next eight rows correspond to Layer C5 single pixel features. The bottom four rows correspond to the Layer 6 outputs corresponding to the four recognition symbols.

"heart" symbol input in Fig. 12. Fig. 13 contains 14 columns. Each corresponds to building frames using the events that appeared during 1 millisecond at some of the ConvNet nodes. Background gray color represents zero events during this millisecond, brighter gray level means a net positive number of events per pixel, while darker gray level means a net negative number of events per pixel. The numbers on the top left of each reconstructed frame indicate the total number of events present for all pixels in that feature map during that millisecond. Each row in Fig. 13 corresponds to one ConvNet node or feature map. The top row corresponds to the 32×32 pixel input crop from the DVS retina. The next six rows correspond to the six subsampled feature maps of the first convolution layer, namely layer S2 14×14 pixel outputs. The next four rows correspond to the 5×5 feature map outputs of layer S4. The next eight rows show the single pixel outputs of layer C5, and the last four rows correspond to the four output pixels of layer 6, each indicating one of the four poker symbol categories. We can see that at the output layer C6 nodes, there is sustained activity for the third row (which corresponds to category "heart") between milliseconds 6 and 12, which is when the input symbol appeared more clearly. During these 6 ms, there were four positive output events for this category, which we artificially have binned into 1 ms slots.

To better illustrate the timing capabilities of multilayer event-driven processing, we selected a 1 ms cut of the input stimulus sequence in Fig. 13. We ran again the simulation for this 1 ms input flash and obtained the results shown in Fig. 14. There are five time diagrams in Fig. 14. The top diagram represents a $(y, time)$ projection of the DVS retina events. Positive events are represented by circles and negative events by crosses. On top of each diagram, we indicate the total number of events in the diagram. The second diagram corresponds to the events in Feature Map 2 of Layer S2. The next diagram represents the events for Feature Map 3 of Layer S4. The next diagram shows the events of all eight neurons in Layer C5, and the bottom diagram shows the events of all neurons in the output Layer C6. We can see that the neuron of category "heart" in Layer C6 provided one positive output event. Fig. 14

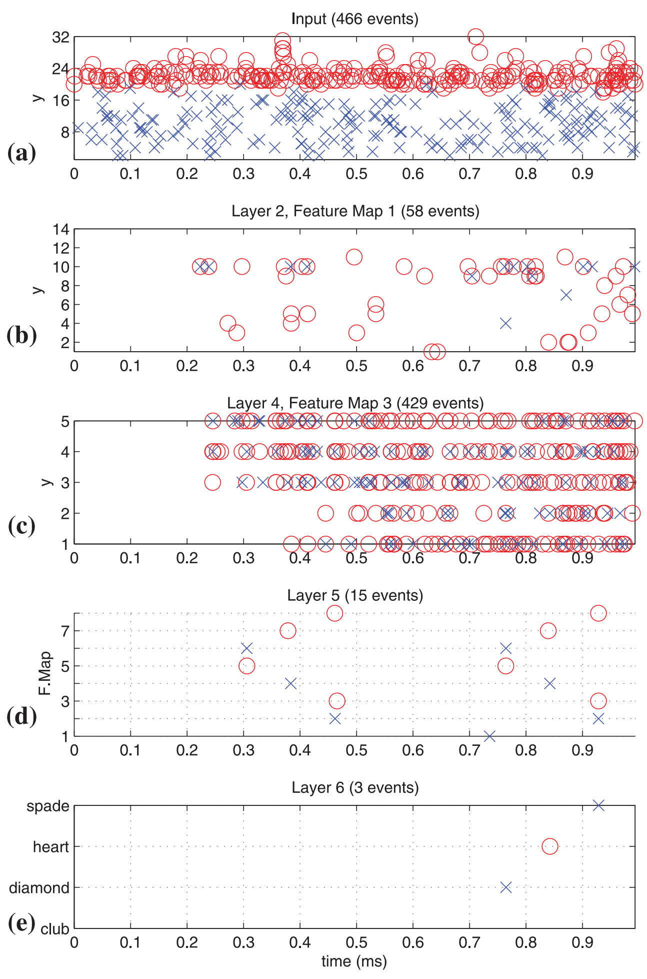


Fig. 14. Events versus time for a simulation that uses as stimulus a 1 ms "heart" symbol cut of that in Fig. 13. Positive events are drawn with circles, while negative events use crosses. (a) 1-ms event flash from DVS 32×32 crop. (b) Events at second feature map FM2 of Layer S2. (c) Events at FM3 of Layer S4. (d) Events at all eight single pixel FMs of Layer C5. (e) Events at all four outputs of Layer C6. Layer C6 produces only three events during this 1-ms "heart" flash. From these three events, only one is positive corresponding to the correct "heart" category.

illustrates nicely the "*pseudosimultaneity*" property or coincidence processing of event-driven multilayer systems. As soon as one layer provides enough events representing a given feature, the next layer feature map tuned to this feature fires events. This property is kept from layer to layer so that output recognition can be achieved while the input burst is still happening.

To characterize the internal representations and dynamics of this event driven network, we show in Appendix 7, available in the online supplemental material, reverse correlation reconstructions for the last layers with down to 0.1-ms time windows.

6 Discussion

Event-driven sensing and processing can be highly efficient computationally. As can be seen from the previous results (for example, Fig. 14), recognition occurs while the sensor is providing events. This contrasts strongly with the conventional frame-driven approach, where the sensor first needs to detect and transmit one image. In commercial video, frame rate is $T_{\mathrm{frame}} = 30$-40 ms. Assuming instantaneous image transmission (from sensor to processor) and instantaneous processing and recognition, the output would therefore be available after T_{frame} of sensor reset (the sensor is reset after sending out a full image). In practice, real-time image processors are those capable of delivering an output at frame rate, that is, after a time T_{frame} of the sensor making an image available or, equivalently, after $2 \times T_{\mathrm{frame}}$ of sensor reset.

Another observation is that in a frame-driven multiconvolution system like the one shown in Fig. 9, the operations in layer n cannot start until operations in layer $n-1$ have concluded. If the system includes feedback loops, then the computations have to be iterated several cycles until convergence for each frame. However, in the event-driven approach, this is not the case. A DVS camera (or any other event-driven sensor) produces output events, while "*reality*" is actually moving (with microseconds delay per event). These events are then processed event by event (with delays of around 100 ns). This effectively makes input and output event flows simultaneous (we have called this *pseudosimultaneity* or *coincidence* property throughout the paper), not only between the input and output of a single event processor but for the full cascade of processors, as in Fig. 9. Furthermore, if the system includes feedback loops, this coincidence property is retained. Recognition delay is therefore not determined by the number of layers and processing modules per layer, but by the statistical distribution of meaningful input events generated by the sensor. Improving the sensor event generation mechanism would thus in principle improve the overall recognition performance and speed of the full system (as long as it does not saturate). For example, improving the contrast sensitivity of a DVS would increase the number of events generated by the pixels for the same stimulus. Also, increasing spatial resolution would multiply the number of pixels producing output events. This way, more events would be generated during the same time, and the "shapes critical for recognition" would become available earlier.

System complexity is increased in a ConvNet by adding more modules and layers. This makes it possible to increase both the "shape dictionary" at intermediate layers and the "object dictionary" at the output layers. However, in an event-driven system, increasing the number of modules per layer would not degrade speed response as long as it does not saturate the communication bandwidth of the intermodule links.

This is one issue to be careful with in event-driven systems. Event traffic saturation is determined by the communication and processing bandwidth of the different AER links and modules, respectively. Each channel in Fig. 9 has a limited maximum event rate communication bandwidth. Similarly, each module also has a maximum event processing rate. Module (filter) parameters therefore have to be set in such a way that maximum communication and processing event rate is not reached. Normally, the event rate is higher for the first stages (sensor and C1), and decreases significantly for later stages. At least this is the case for the feedforward ConvNets we have studied.

One very attractive feature of event-driven hardware is its ease of scalability. Increasing hardware complexity means connecting more modules. For example, with present day high-end ASIC technology, it is feasible to place several large size (64×64 or 128×128) ConvModules (about 10) on a single chip, together with companion routers (to program connectivity) and mappers. An array of 10×10 of these chips can be put on a single PCB, hosting on the order of 1k ConvModules. And then, many of these PCBs could be assembled hierarchically. Several research groups are pursuing this type of hardware assembly goal [27], [28], [30], [45], [52], [53]. In Appendix 6, available in the online supplemental material, we compare in more detail frame versus event-driven approaches focusing on hardware aspects.

Regarding the sensor, we have focused our discussions on the DVS camera. However, there are many reported event-driven sensors for vision and audition. Just to mention a few, there are also plain luminance sensors [31], time-to-first spike coded sensors [32], foveated sensors [33], spatial contrast sensors [34], [35], combined spatiotemporal contrast sensors [36], [37], and velocity sensors [38], [39].

One of the limitations of event-driven hardware compared to frame-driven equipment is that hardware time multiplexing is not possible. For example, present day hardware implementations of frame-driven ConvNets [40] extensively exploit hardware multiplexing by fetching intermediate data in and out between processing hardware and memory. This way, arbitrarily large systems can be implemented by trading off speed. This is not possible in event-driven hardware as events need to "flow" and each module has to hold its instantaneous state.

Another disadvantage of event-driven systems, at least at present, is the lack of efficient, fast training. Spike-time-dependent plasticity (STDP) [41] seems to be a promising unsupervised learning scheme, but it is slow and requires learning synapses with special properties [42]. Other research efforts are dedicated to algorithmic solutions for supervised STDP type learning [19], [20], [21]. However, at the moment, this field is still quite incipient.

Nevertheless, event-driven sensing and processing has many attractive features, and research in this direction is certainly worth pursuing.

7 Conclusions

A formal method for mapping parameters from a frame-driven (vision) neural system to an event-driven system has been presented. Given the extra timing considerations in frame-free event-driven systems, extra degrees of freedom become available. This mapping was illustrated by applying it to example ConvNet systems for recognizing orientations of rotating human silhouettes and fast poker card symbols recorded with real DVS retina chips. The recordings were fed to a hierarchical feedforward spike-driven ConvNet that included 20 event-driven convolution modules. The systems were simulated with a dedicated event-driven simulator. The results confirm the high speed response capability of event-driven sensing and processing systems as recognition is achieved while the sensor is delivering its output.

Acknowledgments

This work was supported by European CHIST-ERA Grant PNEUMA funded by Spanish MICINN (PRI-PIMCHI-2011-0768), Spanish grant (with support from the European Regional Development Fund) TEC2009-10639-C04-01 (VULCANO), and Andalucian Grant TIC609 (NANONEURO).

References

[1] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128x128 120dB 30mW Asynchronous Vision Sensor That Responds to Relative Intensity Change," *Proc. IEEE Int'l Solid-State Circuits Conf.*, 2006.

[2] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128x128 120 dB 15μs Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566-576, Feb. 2008.

[3] J. Kramer, "An Integrated Optical Transient Sensor," *IEEE Trans. Circuits and Systems, Part II*, vol. 49, no. 9, pp. 612-628, Sept. 2002.

[4] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143dB Dynamic Range Asynchronous Address-Event PWM Dynamic Image Sensor with Lossless Pixel-Level Video-Compression," *Proc. IEEE Int'l Solid-State Circuits Conf.*, pp. 400-401, Feb. 2010.

[5] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128x128 1.5% Contrast Sensitivity 0.9% FPN 3us Latency 4mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Amplifiers," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 827-838, Mar. 2013.

[6] F. Gomez-Rodríguez et al., "AER Tools for Communications and Debugging," *Proc. IEEE Int'l Symp. Circuits and Systems*, pp. 3253-3256, May 2006.

[7] E. Chicca, A.M. Whatley, V. Dante, P. Lichtsteiner, T. Delbrück, P. Del Giudice, R.J. Douglas, and G. Indiveri, "A Multi-Chip Pulse-Based Neuromorphic Infrastructure and Its Application to a Model of Orientation Selectivity," *IEEE Trans. Circuits and Systems I, Regular Papers*, vol. 5, no. 54, pp. 981-993, May 2007.

[8] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "A Neuromorphic Cortical-Layer Microchip for Spike-Based Event Processing Vision Systems," *IEEE Trans. Circuits and Systems, Part I: Regular Papers*, vol. 53, no. 12, pp. 2548-2566, Dec. 2006.

[9] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camuñas-Mesa, R. Berner, M. Rivas, T. Delbrück, S.C. Liu, R. Douglas, P. Häfliger, G. Jiménez-Moreno, A. Civit, T. Serrano-Gotarredona, A. Acosta-Jiménez, and B. Linares-Barranco, "CAVIAR: A 45k-Neuron, 5M-Synapse, 12G-Connects/Sec AER Hardware Sensory-Processing-Learning-Actuating System for High Speed Visual Object Recognition and Tracking," *IEEE Trans. Neural Networks*, vol. 20, no. 9, pp. 1417-1438, Sept. 2009.

[10] L. Camuñas-Mesa, A. Acosta-Jiménez, C. Zamarreño-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 32x32 Pixel Convolution Processor Chip for Address Event Vision Sensors with 155ns Event Latency and 20Meps Throughput," *IEEE Trans. Circuits and Systems*, vol. 58, no. 4, pp. 777-790, Apr. 2011.

[11] L. Camuñas-Mesa, C. Zamarreño-Ramos, A. Linares-Barranco, A. Acosta-Jiménez, T. Serrano-Gotarredona, and B. Linares-Barranco, "An Event-Driven Multi-Kernel Convolution Processor Module for Event-Driven Vision Sensors," *IEEE J. Solid-State Circuits*, vol. 47, no. 2, pp. 504-517, Feb. 2012.

[12] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989.

[13] K. Chellapilla, M. Shilman, and P. Simard, "Optimally Combining a Cascade of Classifiers," *Proc. Document Recognition and Retrieval 13*, p. 6067, 2006.

[14] R. Vaillant, C. Monrocq, and Y. LeCun, "Original Approach for the Localisation of Objects in Images," *IEE Proc. Vision, Image, and Signal Processing*, vol. 141, no. 4, pp. 245-250, Aug. 1994.

[15] M. Osadchy, Y. LeCun, and M. Miller, “Synergistic Face Detection and Pose Estimation with Energy-Based Models,” *J. Machine Learning Research,* vol. 8, pp. 1197-1215, May 2007.

[16] C. Garcia and M. Delakis, “Convolutional Face Finder: A Neural Architecture for Fast and Robust Face Detection,” *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 11, pp. 1408-1423, Nov. 2004.

[17] F. Nasse, C. Thurau, and G.A. Fink, “Face Detection Using GPU Based Convolutional Neural Networks,” *Proc. 13th Int'l Conf. Computer Analysis of Images and Patterns,* pp. 83-90, 2009.

[18] A. Frome, G. Cheung, A. Abdulkader, M. Zennaro, B. Wu, A. Bissacco, H. Adam, H. Neven, and L. Vincent, “Large-Scale Privacy Protection in Google Street View,” *Proc. IEEE Int'l Conf. Computer Vision,* 2009.

[19] S.M. Bohte, J.N. Kok, and H. La Poutre, “Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons,” *Neurocomputing,* vol. 48, pp. 17-38, 2003.

[20] O. Booij et al., “A Gradient Descent Rule for Spiking Neurons Emitting Multiple Spikes,” *Information Processing Letters,* vol. 95, no. 6, pp. 552-558, 2005.

[21] F. Ponulak and A. Kasinski, “Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting,” *Neural Computation,* vol. 22, no. 2, pp 467-510, Feb. 2010.

[22] 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.

[23] C. Farabet, B. Martini, P. Akserod, S. Talay, Y. LeCun, and E. Culurciello, “Hardware Accelerated Convolutional Neural Networks for Synthetic Vision Systems,” *Proc. IEEE Int'l Symp. Circuits and Systems,* pp. 257-260, 2010.

[24] http://aerst.wiki.sourceforge.net, 2013.

[25] http://jaer.wiki.sourceforge.net, 2013.

[26] T. Delbrück and P. Lichtsteiner, “Fast Sensory Motor Control Based on Event-Based Hybrid Neuromorphic-Procedural System,” *Proc. IEEE Int'l Symp. Circuits and Systems,* pp. 845-848, 2007.

[27] S. Joshi, S. Deiss, M. Arnold, J. Park, T. Yu, and G. Cauwenberghs, “Scalable Event Routing in Hierarchical Neural Array Architecture with Global Synaptic Connectivity,” *Proc. Int'l Workshop Cellular Nanoscale Networks and Their Applications,* Feb. 2010.

[28] L. Camuñas-Mesa, J.A. Pérez-Carrasco, C. Zamarreño-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco, “On Scalable Spiking ConvNet Hardware for Cortex-Like Visual Sensory Processing Systems,” *Proc. IEEE Int'l Symp. Circuits and Systems,* pp. 249-252, June 2010.

[29] A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, A. Jiménez, M. Rivas, G. Jiménez, and A. Civit, “On the AER Convolution Processors for FPGA,” *Proc. IEEE Int'l Symp. Circuits and Systems,* pp. 4237-4240, June 2010.

[30] R. Silver, K. Boahen, S. Grillner, N. Kopell, and K.L. Olsen, “Neurotech for Neuroscience: Unifying Concepts, Organizing Principles, and Emerging Tools,” *J. Neuroscience,* vol. 27, no. 44, pp. 11807-819, Oct. 2007.

[31] E. Culurciello, R. Etienne-Cummings, and K.A. Boahen, “A Biomorphic Digital Image Sensor,” *IEEE J. Solid-State Circuits,* vol. 38, no. 2, pp. 281-294, Feb. 2003.

[32] S. Chen and A. Bermak, “Arbitrated Time-to-First Spike CMOS Image Sensor with On-Chip Histogram Equalization,” *IEEE Trans. VLSI Systems,* vol. 15, no. 3, 346-357, Mar. 2007.

[33] M. Azadmehr, H. Abrahamsen, and P. Hafliger, “A Foveated AER Imager Chip,” *Proc. IEEE Int'l Symp. Circuits and Systems,* vol. 3, pp. 2751-2754, 2005.

[34] J. Costas-Santos, T. Serrano-Gotarredona, R. Serrano-Gotarredona, and B. Linares-Barranco, “A Spatial Contrast Retina with On-Chip Calibration for Neuromorphic Spike-Based AER Vision Systems,” *IEEE Trans. Circuits and Systems, Part I,* vol. 54, no. 7, pp. 1444-1458, July 2007.

[35] J.A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, “A Five-Decade Dynamic Range Ambient-Light-Independent Calibrated Signed-Spatial-Contrast AER Retina with 0.1ms Latency and Optional Time-to-First-Spike Mode,” *IEEE Trans. Circuits and Systems Part I,* vol. 57, no. 10, pp. 2632-2643, Oct. 2010.

[36] K.A. Zaghloul and K. Boahen, “Optic Nerve Signals in a Neuromorphic Chip I: Outer and Inner Retina Models,” *IEEE Trans. Biomedical Eng.,* vol. 51, no. 4, pp. 657-666, Apr. 2004.

[37] K.A. Zaghloul and K. Boahen, “Optic Nerve Signals in a Neuromorphic Chip II: Testing and Results,” *IEEE Trans. Biomedical Eng.,* vol. 51, no. 4, pp. 667-675, Apr. 2004.

[38] J. Kramer, R. Sarpeshkar, and C. Koch, “Pulse-Based Analog VLSI Velocity Sensors,” *IEEE Trans. Circuits and Systems II,* vol. 44, no. 2, pp. 86-101, Feb. 1997.

[39] C.M. Higgins and S.A. Shams, “A Biologically Inspired Modular VLSI System for Visual Measurement of Self-Motion,” *IEEE Sensors J.,* vol. 2, no. 6, pp. 508-528, Dec. 2002.

[40] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, “Hardware Accelerated Convolutional Neural Networks for Synthetic Vision Systems,” *Proc. IEEE Int'l Symp. Circuits and Systems,* pp. 257-260, 2010.

[41] T. Masquelier, R. Guyonneau, and S. Thorpe, “Competitive STDP-Based Spike Pattern Learning,” *Neural Computation,* vol. 21, pp. 1259-1276, 2009.

[42] G. Indiveri, “Neuromorphic Bistable VLSI Synapses with Spike-Timing-Dependent Plasticity,” *Proc. Advances in Neural Information Processing Systems,* vol. 15, pp. 1091-1098, 2002.

[43] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, “NeuFlow: A Runtime-Reconfigurable Dataflow Processor for Vision,” *Proc. Embedded Computer Vision Workshop,* 2011.

[44] C. Farabet, Y. LeCun, and E. Culurciello, “NeuFlow: A Runtime Reconfigurable Dataflow Architecture for Vision,” *Proc. Snowbird Learning Workshop,* Apr. 2012.

[45] C. Zamarreño-Ramos, A. Linares-Barranco, T. Serrano-Gotarredona, and B. Linares-Barranco, “Multi-Casting Mesh AER: A Scalable Assembly Approach for Reconfigurable Neuromorphic Structured AER Systems. Application to ConvNets,” *IEEE Trans. Biomedical Circuits and Systems,* vol. 7, no. 1, pp. 82-102, Feb. 2013.

[46] C. Farabet, R. Paz, J.A. Pérez-Carrasco, C. Zamarreño-Ramos, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, and B. Linares-Barranco, “Comparison between Frame-Constraint Fix-Pixel-Value and Frame-Free Spiking Dynamic-Pixel ConvNets for Visual Processing,” *Frontiers in Neuromorphic Eng.,* vol. 6, Mar. 2012, doi: 10.3389/fnins.2012.00032.

[47] J. Poulton, R. Palmer, A.M. Fuller, T. Greer, J. Eyles, W.J. Dally, and M. Horowitz, “A 14-mW 6.25-Gb/s Transceiver in 90-nm CMOS,” *IEEE J. Solid-State Circuits,* vol. 42, no. 12, pp. 2745-2757, Dec. 2007.

[48] C. Zamarreño-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco, “An Instant-Startup Jitter-Tolerant Manchester-Encoding Serializer/Deserializar Scheme for Event-Driven Bit-Serial LVDS Inter-Chip AER Links,” *IEEE Trans. Circuits and Systems Part I,* vol. 58, no. 11, pp. 2647-2660, Nov. 2011.

[49] C. Zamarreño-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco, “A 0.35um Sub-ns Wake-Up Time ON-OFF Switchable LVDS Driver-Receiver Chip I/O Pad Pair for Rate-Dependent Power Saving in AER Bit-Serial Links,” *IEEE Trans. Biomedical Circuits and Systems,* vol. 6, no. 5, pp. 489-497, Oct. 2012.

[50] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge Univ. Press, 2002.

[51] C. Zamarreño-Ramos, R. Kulkarni, J. Silva-Martínez, T. Serrano-Gotarredona, and B. Linares-Barranco, “A 1.5ns OFF/ON Switching-Time Voltage-Mode LVDS Driver/Receiver Pair for Asynchronous AER Bit-Serial Chip Grid Links with up to 40 Times Event-Rate Dependent Power Savings,” *IEEE Trans. Biomedical Circuits and Systems* in press.

[52] S.B. Furber, D.R. Lester, L.A. Plana, J.D. Garside, E. Painkras, S. Temple, and A.D. Brown, “Overview of the SpiNNaker System Architecture,” *IEEE Trans. Computers,* doi 10.1109/TC.2012.142, 2012.

[53] E. Painkras, L.A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D.R. Lester, A.D. Brown, and S.B. Furber, “SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation,” *IEEE J. Solid-State Circuits,* vol. 48, no. 8, pp. 1943-1953, Aug. 2013.

José Antonio Pérez-Carrasco received the degree in telecommunication engineering in 2004, and the PhD degree from the University of Seville, Spain, in March 2011. He started collaborating with the Biomedical Image Processing Group (BIP) in 2003, when he was working on his BSc thesis under the supervision of the BIP group leaders Dr. Serrano and Dr. Acha. After working in industry, in 2005 he received two 1-year research grants to implement vision processing algorithms within the Sevilla Microelectronics Institute in collaboration with the BIP group at the University of Seville. In 2007, he received the PhD grant. He is currently an assistant professor in the Department of Signal Theory, University of Seville. His research interests include visual perception, real-time processing, pattern recognition, image processing, and its medical applications.

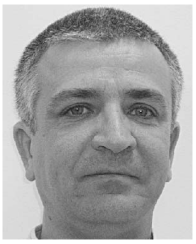
Bo Zhao received the BS and MS degrees in electronic engineering from Beijing Jiaotong University, China, in 2007 and 2009, respectively. He is currently working toward the PhD degree in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include design and VLSI implementation of bioinspired vision processing algorithms. He is a student member of the IEEE.

Carmen Serrano received the MS degree in telecommunication engineering from the University of Seville, Spain, in 1996 and the PhD degree in January 2002. In 1996, she joined the Signal Processing and Communication Department at the same university, where she is currently a tenured professor. Her research interests include image processing and, in particular, color image segmentation, classification, and compression, mainly with biomedical applications. She is a member of the IEEE.

Begoña Acha received the PhD degree in telecommunication engineering in July 2002. Since 1996, she has been working in the Signal Processing and Communications Department, University of Seville, where she is currently a tenured professor. Her current research activities include works in the field of color image processing and its medical applications. She is a member of the IEEE.

Teresa Serrano-Gotarredona received the BS degree in electronic physics and the PhD degree in VLSI neural categorizers from the University of Sevilla, Spain, in 1992 and 1996, respectively, and the MS degree in electrical and computer engineering from The Johns Hopkins University, Baltimore, Maryland, in 1997. She was an assistant professor in the Electronics and Electromagnetism Department, University of Sevilla, from September 1998 to September 2000. Since September 2000, she has been a tenured scientist at the National Microelectronics Center (IMSE-CNM-CSIC), Sevilla, Spain, and in 2008, she was promoted to tenured researcher. Her research interests include analog circuit design of linear and nonlinear circuits, VLSI neural-based pattern recognition systems, VLSI implementations of neural computing and sensory systems, transistor parameters mismatch characterization, address-event-representation VLSI, RF circuit design, nanoscale memristor-type AER, and real-time vision processing chips. She is the coauthor of the book *Adaptive Resonance Theory Microchips* (Kluwer, 1998). She has been an associate editor of the *IEEE Transactions on Circuits and Systems-Part I, Regular Papers* since December 2011, and an associate editor for *PLoS ONE* since 2008.

Shouchun Cheng received the BS degree from Peking University, the ME degree from the Chinese Academy of Sciences, and the PhD degree from the Hong Kong University of Science and Technology in 2000, 2003, and 2007, respectively. He held a postdoctoral research fellowship in the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology for one year after graduation. From February 2008 to May 2009, he was a postdoctoral research associate in the Department of Electrical Engineering, Yale University. In July 2009, he joined Nanyang Technological University as an assistant professor. He serves as a technical committee member of Sensory Systems, IEEE Circuits and Systems Society, an associate editor of *IEEE Sensors Journal*, an associate editor of the *Journal of Low Power Electronics and Applications*, a program director (Smart Sensors) of VIRTUS, IC Design Centre of Excellence, a regular reviewer for a number of international conferences and journals such as *TVLSI*, *TCAS-I/II*, *TBioCAS*, *TPAMI*, *Sensors*, *TCSVT*, and so on. He is a member of the IEEE.


Bernabé Linares-Barranco received the BS degree in electronic physics, the MS degree in microelectronics, and the PhD degree in high-frequency OTA-C oscillator design from the University of Sevilla, Spain, in 1986, 1987, and 1990, respectively, and the PhD degree in analog neural network design from Texas A&M University, College Station, in 1991. Since September 1991, he has been a tenured scientist at the Sevilla Microelectronics Institute, which is one of the institutes of the National Microelectronics Center of the Spanish Research Council of Spain. In January 2003, he was promoted to tenured researcher and, in January 2004, to a full professor of research. Since March 2004, he has also been a part-time professor at the University of Sevilla. From September 1996 to August 1997, he was on sabbatical in the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, Maryland, as a postdoctoral fellow. During spring 2002, he was a visiting associate professor in the Electrical Engineering Department, Texas A&M University. He is the coauthor of the book *Adaptive Resonance Theory Microchips* (Kluwer, 1998). He was also the coordinator of the EU-funded CAVIAR project. He has been involved with circuit design for telecommunication circuits, VLSI emulators of biological neurons, VLSI neural-based pattern recognition systems, hearing aids, precision circuit design for instrumentation equipment, bioinspired VLSI vision processing systems, transistor parameter mismatch characterization, address-event-representation VLSI, RF circuit design, real-time vision processing chips, and extending AER to the nanoscale. He was a corecipient of the 1997 *IEEE Transactions on Very Large Scale Integration Systems* Best Paper Award for the paper "A Real-Time Clustering Microchip Neural Engine." He was also corecipient of the 2000 IEEE Circuits and Systems Darlington Award for the paper "A General Translinear Principle for Subthreshold MOS Transistors." From July 1997 until July 1999, he was an associate editor of the *IEEE Transactions on Circuits and Systems—Part II, Analog and Digital Signal Processing*, and from January 1998 to December 2009, he was an associate editor for the *IEEE Transactions on Neural Networks*. He has been an associate editor of *Frontiers in Neuromorphic Engineering* since May 2010. He was the chief guest editor of the 2003 *IEEE Transactions on Neural Networks* special issue on neural hardware implementations. From June 2009 until May 2011, he was the chair of the Sensory Systems Technical Committee of the IEEE CAS Society. In March 2011, he became a chair of the IEEE CAS Society Spain Chapter.