An Event-Driven Multi-Kernel Convolution Processor Module for Event-Driven Vision Sensors

Luis Camuñas-Mesa, Carlos Zamarreño-Ramos, Alejandro Linares-Barranco, *Member, IEEE*, Antonio J. Acosta-Jiménez, Teresa Serrano-Gotarredona, *Member, IEEE*, and Bernabé Linares-Barranco, *Fellow, IEEE*

***Abstract*—Event-Driven vision sensing is a new way of sensing visual reality in a frame-free manner. This is, the vision sensor (camera) is not capturing a sequence of still frames, as in conventional video and computer vision systems. In Event-Driven sensors each pixel autonomously and asynchronously decides when to send its address out. This way, the sensor output is a continuous stream of address events representing reality dynamically continuously and without constraining to frames. In this paper we present an Event-Driven Convolution Module for computing 2D convolutions on such event streams. The Convolution Module has been designed to assemble many of them for building modular and hierarchical Convolutional Neural Networks for robust shape and pose invariant object recognition. The Convolution Module has multi-kernel capability. This is, it will select the convolution kernel depending on the origin of the event. A proof-of-concept test prototype has been fabricated in a 0.35μ m CMOS process and extensive experimental results are provided. The Convolution Processor has also been combined with an Event-Driven Dynamic Vision Sensor (DVS) for high-speed recognition examples. The chip can discriminate propellers rotating at 2 k revolutions per second, detect symbols on a 52 card deck when browsing all cards in 410 ms, or detect and follow the center of a phosphor oscilloscope trace rotating at 5 KHz.**

***Index Terms*—Address-event representation (AER), asynchronous vision sensors and processors, high-speed imaging, image convolutions, image sensors, machine vision, neural networks hardware, neuromorphic circuits, robot vision systems, visual system.**

I. Introduction

STATE-OF-THE-ART artificial vision technology is presently based on capturing and processing a sequence of still image frames. Present day development trends are towards more pixels per frame to increase spatial resolution and higher frame rates to increase temporal resolution. Unfortunately, this trend also yields exponentially increasing data rate, thus imposing higher computational and power demands on later processing stages.

However, in the last years we have been witnessing a new type of vision sensors appearing in the specialized literature, which are not based on capturing sequences of frames. Taking inspiration from biological retinas, the pixels in these sensors decide when sending information out, as opposed to waiting for an externally controlled periodic sampling instant. These sensors are said to be "Event-Driven", because each pixel sends out an information event when it senses a given level of luminance, or a level of spatial or temporal contrast. Example Event-Driven sensors are: simple luminance to frequency transformation sensors [1], time-to-first event coding sensors [2]–[4], foveated sensors [5], [6], temporal contrast vision sensors [7]–[11], motion sensing and computation systems [12], [13], and spatial contrast sensors [9], [10], [14]–[16], among many others. Of special interest for very high-speed applications are the so-called "Dynamic Vision Sensors" (DVS), where each pixel autonomously computes the normalized time derivative of the sensed light ($\dot{I}/I$) and provides an output event with its (x, y) coordinate when this amount exceeds a preset contrast [7], [8], [11]. Fig. 1 illustrates the operation of such sensor when observing a 7.5 KHz spiral on an analog oscilloscope operated in x-y mode and stimulated with two phase shifted sinusoids of decreasing amplitude. The only ambient light is that of the oscilloscope phosphor. Fig. 1(b) represents in 3D (x, y, t) coordinates the address events produced by the sensor during about 500 μs. The sensor provides events with a latency of a few microseconds to milliseconds depending on illumination, but relative inter-event temporal resolution is in the order of hundreds of nano seconds. Thus, it is possible to know when the oscilloscope spot crossed a pixel with sub-microsecond timing precision. In general, Event-Driven Sensors share interesting properties such as fast sensing capability, reduced information throughput, low power, and efficient in-sensor pre-processing, which makes them attractive for low power portable applications as well as high-speed scenarios.

But sensing is simply the first step in a vision system. The next step is performing processing to achieve a desired functionality, such as object recognition. As these Event-Driven sensors provide information events with sub-microsecond time resolution, it is obvious that the most efficient way of processing would be event by event, as opposed to histogramming them into artificial frames and use conventional frame-based image processing techniques. Taking further inspiration from biology,

This work was supported by EU Grant 216777 (NABAB), and Spanish Grant TEC2009-10639-C04-01 (VULCANO) with support from the European Regional Devel-opment Fund. The work of C. Zamarreño-Ramos was supported by a national FPU scholarship.

L. Camuñas-Mesa is with the Department of Engineering, University of Leicester, U.K.

C. Zamarreño-Ramos is with the Sevilla Microelectronics Institute (IMSE-CNM-CSIC), Sevilla 41092, Spain.

A. Linares-Barranco is with the Department of Computer Architecture and Technology, University of Sevilla, Sevilla 41092, Spain.

A. Acosta-Jiménez is with the Sevilla Microelectronics Institute (IMSE-CNM-CSIC), Sevilla 41092, Spain, and also with the Department of Electronics and Electromagnetism, University of Sevilla, Sevilla 41092, Spain.

T. Serrano-Gotarredona and B. Linares-Barranco are with the Sevilla Microelectronics Institute (IMSE-CNM-CSIC), Spain, and also with the Department of Computer Architecture and Technology, University of Sevilla, Spain.

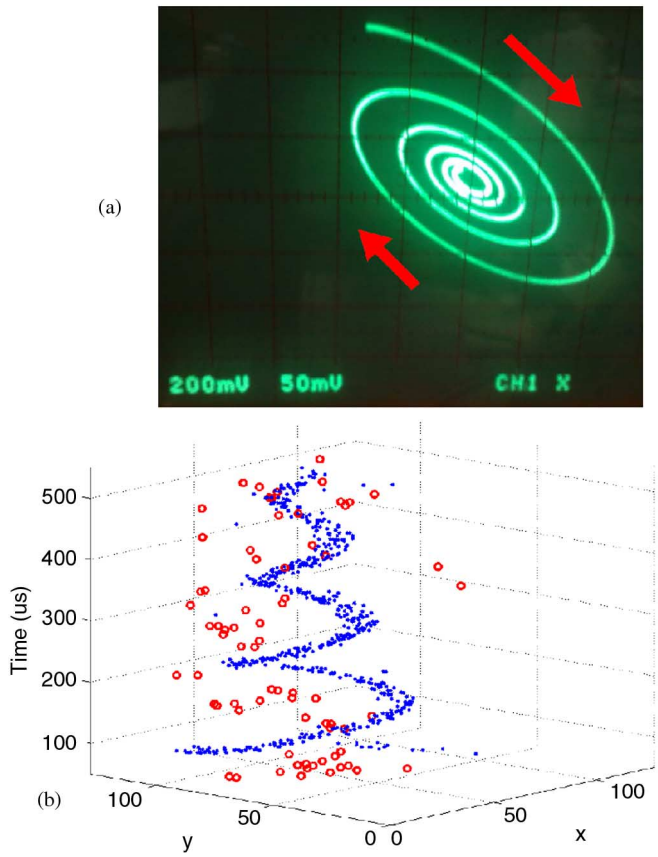

Fig. 1. Example of DVS (Dynamic Vision Sensor) high-speed Event-Driven sensing. (a) The sensor is observing a 7.5 KHz spiral on an analog oscilloscope with no other ambient light than the screen phosphor. (b) Address Events captured by the DVS during 500 μm represented in 3D (x, y, t) space. Dots are fast OFF-ON positively signed events, while circles are slow ON-OFF negatively signed events.

these frame-free event flows can be fed to "*Event-Driven Feature Extractor Modules.*" This way, as in the brain [17], a first layer would extract low level features such as short oriented edges at different scales and angles, a second layer would combine them into more complex shapes, and subsequent layers would continue to combine simpler features into more complex and specialized ones, until recognizing specific objects. Such computational systems have been studied intensively in the fields of computational neuroscience and artificial vision [18]–[20]. Many of them can be described by the computational paradigm known as "*Convolutional Neural Networks*" or *ConvNets* in brief. ConvNets take direct inspiration from Hubel and Wiesel Nobel Prize winning discoveries [17]. Their first software implementation goes back to 1969 [21], with strong developments since the 90s starting with the work at AT&T Bell Labs [22]. Today, ConvNets count with many successful commercial applications such as AT&T/Lucent-Technologies/NCR check reading ATM machines [23], Microsoft OCR and handwritten character recognition systems [24], Thomson developments in face/object recognition [25], France Telecom/Orange with face detection and recognition, as well as text detection and recognition [26]–[28], or Google developments for detecting faces and license plates in StreetView images [29]. So far, ConvNets have been developed for frame-based sensors, mostly as software programs, although some digital hardware implementations have been reported [30]–[35]. In this paper we present an Event-Driven Convolutional Module (ConvModule) intended as a starting building block for assembling expandable, modular, hierarchical and reconfigurable ConvNet vision systems for Event-Driven Vision Sensors.[1] These ConvModules process the event flow coming from the sensors event by event, with processing delays in the range of a hundred nano seconds, thus preserving the microsecond time resolution of the sensor events. Also, as input and output event flows have negligible per-event processing delays, the two flows are virtually simultaneous. This is a fundamental difference with respect to frame-based sensing and processing systems, where the delay of the combined sensor + processor systems can never be smaller than frame rate.

The paper is structured as follows. In the next Section we quickly review the generic concept of Event-Driven convolution operation and summarize previous work as well as motivation for the present design. Section III describes the operation of the presented ConvModule and introduces the architectural and circuit changes with respect to previous designs. Section IV provides experimental characterization results of a fabricated chip prototype, Section V discusses on scalability and future outlook, Section VI compares with some frame-based GPU/FPGA implementations, and finally Section VII draws conclusions.

II. Event-Driven Convolution Operation and Previous Work

Fig. 2 illustrates the operation of an Event-Driven ConvModule (chip) when fed by events produced by an Event-Driven vision sensor chip (like those reported by Delbrück [7], Posch [11], or Leñero-Bardallo [8]). On the left, there is an Event-Driven Vision Sensor chip, and on the right an Event-Driven processor like the one reported in this paper. In the Event-Driven Sensor chip, each pixel includes a photo sensor and some pre-processing circuitry to compute, for example, spatial or temporal contrast. Whenever a pixel detects a given level of contrast, it requests access to the AER (Address Event Representation) bus to send out its (x, y) coordinate using asynchronous handshaking [36]. The Event-Driven ConvModule (chip) receives this event and sends it to a "*Projection Field*" of pixels. Each pixel in the ConvModule accumulates the contributions of the incoming events, until reaching a threshold, in which case it will send out a new event through the ConvModule output AER port. If $(x_{\rm in}, y_{\rm in})$ is the coordinate of the incoming event and $(x_{\rm c}, y_{\rm c})$ is a pixel within the *Projection Field* of this event, the contribution of the input event to the projection field pixels is weighted by a factor w which depends on their relative spatial positions $w = w(x_{\rm c} - x_{\rm in}, y_{\rm c} - y_{\rm in})$. The 2D function $w(x, y)$ defines the convolution kernel. By accumulating the continuous flow of input event kernel-weighted contributions, the output flow of events represents the 2D convolution of the input flow with kernel $w(x, y)$ [37].

[1] Although reported event-driven ConvModule prototypes always occupy one full chip, as in the present paper prototype, the ultimate goal would be to integrate many of them in a NoC (Network on Chip) die, together with routing and reconfiguration engines. But this is beyond the scope of the present paper. Section V provides a futuristic outlook of what might be expected.

TABLE I
CONVMODULE CHIPS PERFORMANCE COMPARISON

	Venier [38]	Choi [39]	Serrano [40]	Camuñas [41]	Present Work
Technology	2μm	0.25μm	0.35μm	0.35μm	0.35μm
Chip size	7.2x6.1mm^2	3.8x2.5mm^2	4.0x5.0mm^2	4.3x5.4mm^2	5.5x5.8mm^2
pixel array	27x27	32x64	32x32	32x32	64x64
pixel size	205x160μm^2	52x49μm^2	93x96μm^2	96x101μm^2	58x54μm^2
max kernel size	10x10	32x32	32x32	32x32	32x32
arbitrary kernel	no	no	yes	yes	yes
max kernels	1	1	1	1	32
event processing time	10μs	N/A	30-340ns	50-565ns	60-680ns
computation	analog	analog	analog	digital	digital
precision	N/A	N/A	3-bit	18-bit	6-bit
latency in-to-out	N/A	N/A	1ms	155ns	175ns
power dissipation	<9mW	6mW$_{max}$ [i]	150mW$_{max}$	200mW$_{max}$	200mW$_{max}$

i. Measured by stimulating one pixel only.

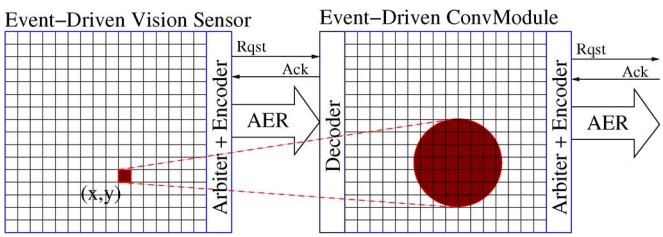

Fig. 2. Illustration of Event-Driven Convolution Processing. When an event of coordinate (x, y) is received from an event-driven vision sensor, a contribution is sent to a "Receptive Field" of pixels in the destination Event-Driven ConvModule.

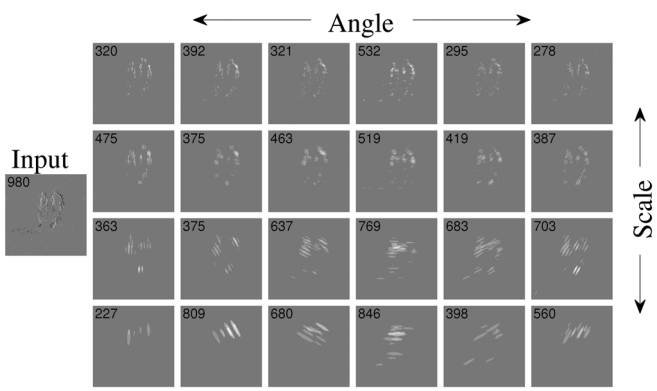

Fig. 3. Array of 24 Event-Driven Gabor filters. Each subfigure corresponds to events captured during the same 40 ms time window, and each number in the subfigure indicates the number of events captured during this 40 ms time window. The subfigure on left margin is obtained from a DVS retina sensor. The other 24 subfigures correspond to Event-Driven Convolution Filtering operations using Gabor kernel of six different orientations and four different scales. The input and all 24 outputs are simultaneous.

Fig. 3 shows an example simulation of AER flow convolution processing on real sensory data. The event flow generated by a 128×128 pixel motion detection retina is sent in parallel to an array of 24 ConvModules, each programmed with a Gabor kernel of different scale and orientation. The retina is looking at two persons walking and providing events representing their moving silhouettes. These events are sent to all 24 ConvModules in parallel, which compute the events representing a 2D Gabor filter operation of the input. Each sub-figure in Fig. 3 corresponds to assembling a 2D histogram by collecting the events during the same 40 ms time window. The events coming out of the sensor have a typical delay of a few microseconds to milliseconds with respect to reality, depending on ambient light [7], [8], [11]. The ConvModules need about 100 ns to 1 μm to process each event, depending on kernel size. Each ConvModule needs to collect a given number of space-time correlated input events to provide an output event, depending on the ConvModule settings. For high-speed processing, one can set this number to be around ten events or less. In general, more relevant pixels in the sensor have stronger signals and send out their events sooner or more frequently. Consequently, more relevant events will be processed first by later ConvModules. This way, in an object recognition hierarchical ConvNet, recognition can be achieved as soon as the sensor provides enough significant space-time correlated events. We refer to this as the "*pseudo-simultaneity*" property of event-driven convolution processing. This input-to-output *pseudo-simultaneity* property of event-driven convolutions is very attractive for hierarchical multi-layer ConvNets in object recognition applications. One key requirement, however, is that the sensor provides a sparse representation of the observed reality in order to not saturate the finite peak event rate of the AER links. In general, each ConvModule reduces the event rate from input to output. Therefore, typically the highest event rate is found at the sensor output. In Fig. 3, the input flow contains 980 events for the selected 40 ms time window. The number of events produced by the ConvModules during the same 40 ms varies from 227 to 846 events.

Table I summarizes the features of previously reported Event-Driven ConvModule chips. The first event-driven ConvModule chip was reported in 1997 by Venier *et al.* [38]. It implemented a hard-wired elliptic kernel, whose shape could be fine tuned through analog biases. In 1999, Serrano *et al.* reported an architecture to perform 2D event-driven convolutions with kernels of more generic shape, as long as

the kernel could be decomposed into a horizontal and vertical components $w(x,y) = H(x)V(y)$ [37]. In 2005 Choi *et al.* [39] presented an event-driven ConvModule chip for Gabor-like kernels, where again the shape of the kernels could be fine tuned through analog biases. In 2006, Serrano *et al.* presented an event-driven ConvModule chip for generic kernels of arbitrary shape and size [40], where kernels were uploaded on a kernel-RAM at start-up. All these event-driven ConvModule chips used analog charge packet integration on in-pixel capacitors to perform weighted event integration. Given the severe area and current consumption restrictions required per pixel, analog computing circuits were designed to operate in weak inversion. Consequently, they suffered from severe transistor mismatch and low computational precision. In particular, Serrano's chip which included an in-pixel calibration circuitry to improve precision, only managed to reach an overall 3-bit precision. The limited precision, tedious calibration process, and critical analog biasing motivated the exploration of a fully digital pixel using digital adders and accumulators to perform the weighted event integration. A preliminary prototype was developed using conservative digital circuit techniques and an oversized 18-bit precision for the pixel accumulation registers [41]. Another severe limitation of all previously reported event-driven ConvModule prototypes was that they could only operate a single-kernel convolution. However, for generic ConvNet systems assembly, each module receives several visual input flows and requires to compute and accumulate a convolution with different kernel for each. Fig. 4 shows the multi-layer feed-forward structure of a typical ConvNet architecture [21]–[31]. Typically, there are between three to eight sequential layers. Each layer contains a set of "*Feature Maps*" (FM). Each FM in the first layer after the sensor receives input from the sensor only and computes a single kernel convolution. However, FMs starting from the second layer receive more than one visual flow, and for each have to use a different kernel to compute and accumulate the convolutions. The event-driven ConvModule we present in this paper has multi-kernel capability and can compute and accumulate different kernel convolutions in parallel for multiple simultaneous input flows. Also, the pixel accumulators use a more realistic register size of 6 bits while using more compact pass transistor logic circuits. As a result, pixel area is approximately one fourth of the previous design, thus allowing higher pixel density. Additionally, it allows for up to 32 separate kernels. Performing 32 kernel convolutions with prior ConvModules, requires the use of 33 of them: one for each kernel plus an extra one for adding all 32 outputs.

III. ConvModule Description

The architecture of the ConvModule is shown in Fig. 5(a). It contains a synchronous controller with an internal clock, a 32×32 4-bit words static kernel-RAM, a kernel parameter lookup table (LUT), a column blocker, a 2's complement block, a left/right column shifter, an array of 64×64 pixels, and an asynchronous event read out block which follows row-parallel burst-mode event read-out [42].

Event-driven convolutions are performed as follows [37]–[41]. Pixels (x,y) in the "Pixel Array" (see Fig. 5(a)) hold their state in a continuous and dynamic manner. When the module receives an input event $(x_{\text{in}}, y_{\text{in}}, s_{\text{in}}, k_{\text{in}})$, kernel k_{in} (which is a 2D matrix stored in the "Kernel-RAM") is added/subtracted to the "Projection Field" of pixels around "Event Address" $(x_{\text{in}}, y_{\text{in}})$, as illustrated in Fig. 5(a). Input event sign s_{in} determines whether the kernel is added or subtracted. Independently of the input event flow, all pixels "suffer" from a constant rate leak that will drive their state to a resting level. When a pixel reaches a positive (negative) threshold, it is reset to its resting level, and a positively (negatively) signed output event is sent out through the AER output port with the pixel's coordinate. This way, ConvModules are excellent spatio-temporal feature extractors, because if enough input events representing the kernel spatial feature are received close in time (to avoid the effect of the leak), output events representing the location of these features are produced.

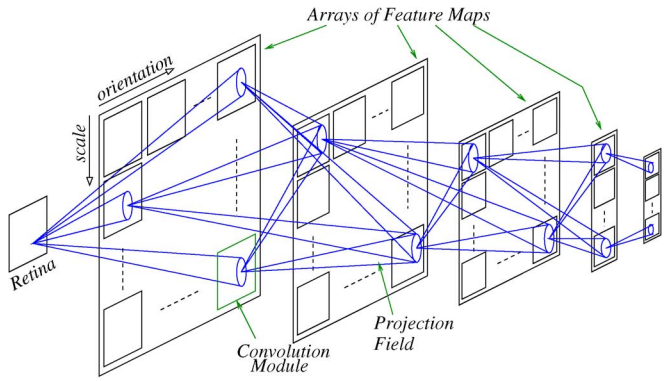

Fig. 4. Typical hierarchical structure of a feed forward ConvNet architecture.

To perform these simplistic conceptual operations, the machinery described below is used. To devise this machinery, the main strategic criterion was to perform each event kernel addition/subtraction as fast as possible. For this, kernel lines are copied in one controller clock-cycle from the kernel-RAM to the Pixel Array, and added-accumulated during the next clock-cycle. Another strategic choice was to fully decouple the output events read-out process from the input events convolution computations, by having the synchronous controller take care of this last process only. Next, we describe in more detail the different operations.

A. Synchronous Controller

The controller is outlined in Fig. 5(b). At start-up, the kernels are loaded onto the static kernel-RAM and other necessary kernel parameters onto the kernel-LUT. Also, registers that store configuration and control parameters are loaded at start-up. Input asynchronous events are fed in through synchronizers into a small 4-position FIFO. Then a Finite-State-Machine (FSM), described in VHDL, controls the sequential operation of copying appropriately the selected kernel onto the desired pixels within the pixel array. Each incoming event includes: 1) a 14-bit event address (x,y); 2) an event sign bit; and 3) a 5-bit kernel number to select one of up to 32 stored kernels. Although the present ConvModule has only 64×64 pixels, it can "see" a 128×128 input space. Configuration parameters $(i,j)_{min}$ and $(i,j)_{max}$ define the pixel array position within

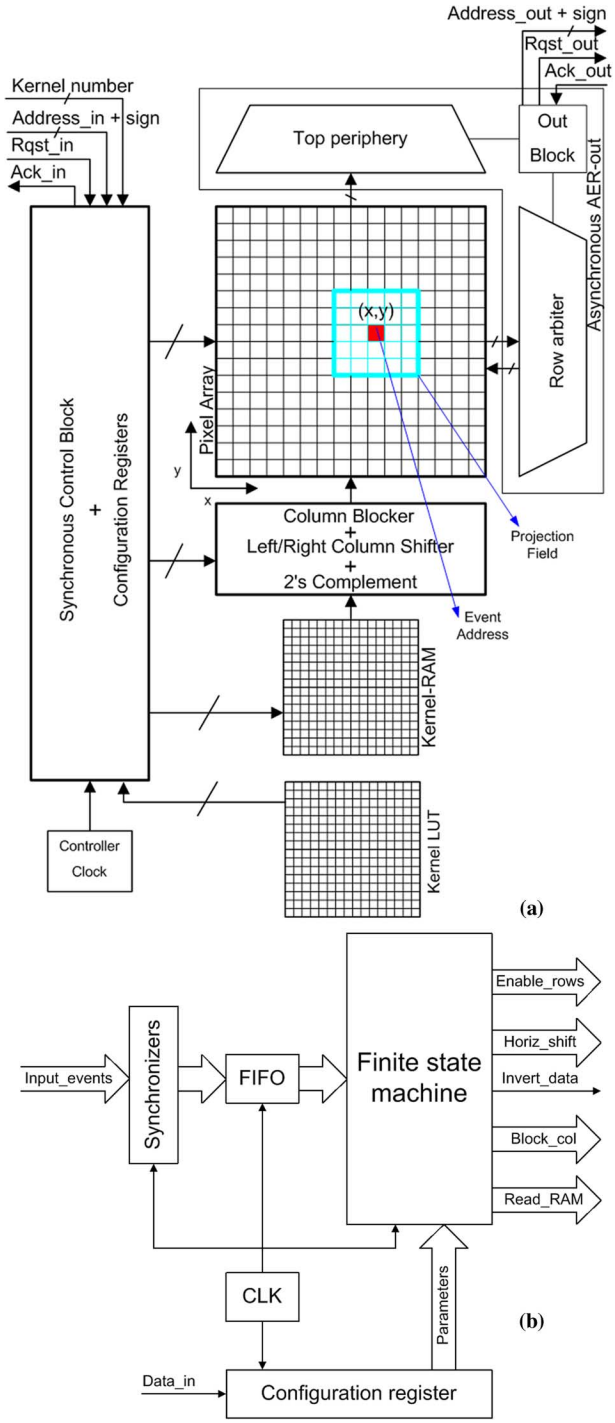

Fig. 5. (a) ConvModule architecture. (b) Controller block diagram.

the total 128×128 input space. This allows to assemble several ConvModules in parallel, each processing a different 64×64 tile of the total 128×128 space. For processing larger spaces, an extra address remapper module [44] would be required for each 128×128 tile.

B. Controller Finite State Machine Description

Fig. 6 shows a simplified state transition diagram of the synchronous controller FSM. By default it stays in a resting state waiting for a new input event. Once an event is received and the

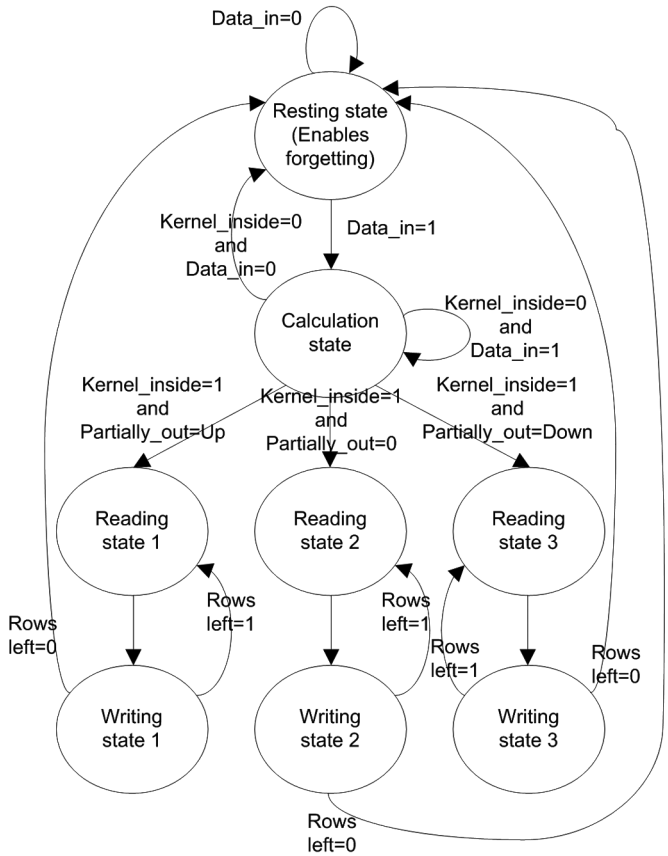

Fig. 6. Synchronous Controller FSM simplified state transition diagram.

kernel selected, the first computation is to detect whether the selected kernel lies fully inside the pixel array, fully outside, or whether some rows fall outside either from the bottom or the top. If the kernel lies fully outside, the operation for this event is finished, and the FSM returns to the resting state. In the other cases, the FSM computes which kernel-RAM rows need to be copied onto which pixel array rows and copies them sequentially one after the other. Depending on the situation, these computations differ, and the FSM follows three possible state paths: 'Reading/Writing state2' if the kernel lies fully inside the pixel array, 'Reading/Writing state1' if the kernel top rows fall outside the pixel array, and 'Reading/Writing state3' if the bottom kernel rows fall outside. For each row, all values are copied in parallel. Since there is more than one kernel in the kernel-RAM, the "*Block_col*" signal in Fig. 5(b) activates through the "*Column blocker*" only those columns where the selected kernel is stored. If the input event has negative sign, the 2's complement block is activated to invert the sign of all weights of the selected kernel, through FSM signal "*invert_data*". FSM signal "*Horiz_shift*" controls a switch matrix (left/right column shifter) to align the kernel-RAM columns where the selected kernel is stored to the pixel array columns where it needs to be copied. After a secured delay, the FSM enables the destination pixel array row, so that the active kernel row is copied onto the corresponding pixel array row. This row copy operation is sequentially repeated for all required kernel rows. The controller needs $n_{clk} = 4 + 2n_k$ clock cycles, where n_k is the number of kernel

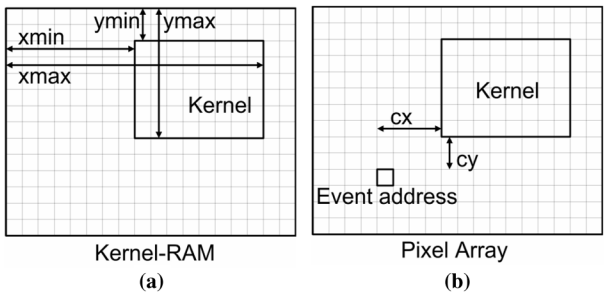

Fig. 7. Kernel parameters definition. (a) Kernel position and size within Kernel-RAM and (b) kernel position with respect to input event coordinate within pixel array space.

rows to copy. For each kernel stored, we need six extra parameters, besides the values in the kernel-RAM. Fig. 7(a) shows one kernel stored inside the 32×32 4-bit words kernel-RAM and the four 5-bit parameters needed to define the kernel location and size $\{xmin, xmax, ymin, ymax\}$. Also, in general, the kernel might not be centered with respect to the event address, but could be displaced by cx and cy, as shown in Fig. 7(b). Every time an input event is received, the FSM loads these six parameters from the kernel-LUT for the input event kernel number and computes the sequence of FSM output control signals appropriately. The rest of parameters stored in the configuration registers are for (a) selecting internal or external clock, (b) enabling/disabling leak, (c) selecting accumulator limit, (d) selecting whether positive or negative events are discarded (thus reducing AER bus traffic if they will not be used), (e) setting the leak rate accumulator limit, and (f) parameters $(i,j)_{min}$ and $(i,j)_{max}$ which define the position of the 64×64 pixel array within the total 128×128 visible input space.

C. Leak/Forgetting Capability

Besides the kernel row-wise copy operation, the controller also generates a periodic signal for all pixels in the array, to perform a constant rate leak of the accumulated value. This is a necessary feature in Event-Driven frame-free processing systems. Note that a ConvModule provides output events at specific coordinates if a set of space-time correlated input events represent the "feature" defined by the kernel. Consequently, these correlated input events have to be received during a time interval, controlled by the leak-rate. This leak-rate is adjusted to the time constant of the "*reality*" under observation. If the sensor is observing walking humans, this rate should be adjusted to be in the range of tens to hundreds of milliseconds. If it is used in a microscopic particle tracking application [45], then the required time constant should be rather in the microsecond range.

D. Pixel Description

Fig. 8 shows the block diagram of the pixel. Kernel weights w_{ij} (3-bit plus sign) are received column-wise, and added/subtracted to a 6-bit accumulator, using 2's complement logic. Row-wise signal "*Enable*" comes from the FSM and enables kernel addition of only one row of the pixel array. Signal "*Sel_forgetting*" activates addition/subtraction of a fixed integer for all pixels in the array, to perform the constant rate leak. Leak addition or subtraction is selected depending on the pixel accumulator sign. The accumulator is monitored by a comparison block that detects whether a positive or negative threshold is reached. If reached, the accumulator is reset to zero, and the pixel requests to send out a signed output event. The in-pixel AER interface is standard for row-parallel event read-out [42]. Fig. 9 shows the schematic of the 16-transistor adder cell used. It employs pass transistor logic with buffering of critical nodes.

TABLE II
CHIP SPECIFICATIONS

Technology	4M 2P 0.35 μm CMOS
Pixel size	$58.0 \times 53.8 \mu m^2$
Chip size	$5.5 \times 5.8 mm^2$
Pixel array	64×64
Pixel computing resolution	6 bits
Kernel resolution	4 bits
Signed computations	Yes
Multikernel system	Yes
Events inhibition	Yes
Input event throughput	1.47-16.6 Meps
Max. output event rate	37 Meps
Min. latency in-to-out	175ns
Power consumption	200mW max

IV. EXPERIMENTAL RESULTS

A test prototype chip holding the ConvModule has been fabricated in the AMS 0.35 μm CMOS technology with 3.3 V power supply. A chip microphotograph is shown in Fig. 10 and specifications are summarized in Table II. It occupies 5.5×5.8 mm^2 with each pixel using 58.0×53.8 μm^2. A close-up of the pixel layout is shown in Fig. 11 highlighting the main parts. Pixel array is 64×64 with 6-bit adder/accumulator, kernel-RAM array size is 32×32 with 4-bit 2's complement words. Controller clock operates at 100 MHz. Chip power consumption depends greatly on input and output event traffic, and an important fraction is consumed by the output event pads, reaching a maximum of about 200 mW.

A. ConvModule Timing Characteristics

Fig. 12 shows a characteristic timing diagram of input and output handshaking signals. Kernel weights and pixel thresholds were set so that each input event would generate ten output events, coming from five consecutive rows. Output $Rqst$ and Ack signals are shorted. Events coming from the same row are read out at a rate of 27 ns per event, while when switching to a new row requires 60 ns. Measured input to output latency is 192 ns. Maximum sustained input event rate depends on the number of kernel lines n_k, since the controller needs $n_{clk} = 4 + 2n_k$ clock cycles for processing one event. Therefore, input event rate varies between 1.47 and 16.6 Meps (mega events per second) depending on n_k, while output event rate can reach up to 37 Meps.

B. Illustration of Luminance Processing

AER is highly inefficient when using rate coding to represent a static luminance image, because the total number of events becomes excessively large. This is why, in practical AER vision

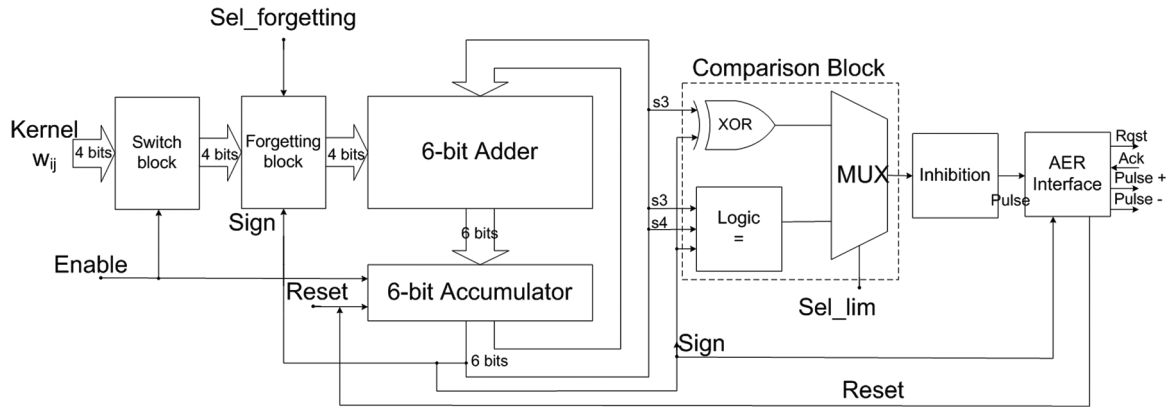

Fig. 8. Pixel block diagram.

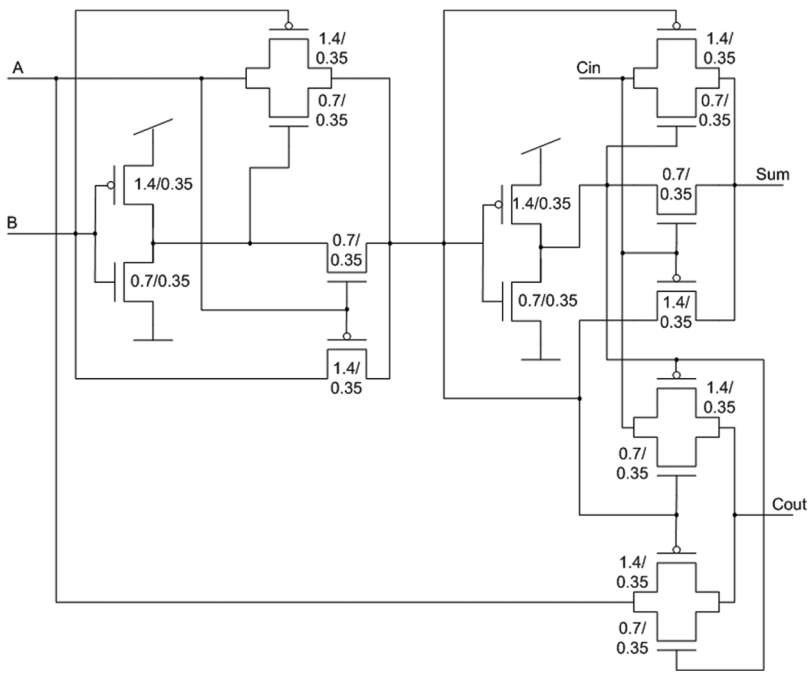

Fig. 9. Pixel 16-transistor adder cell.

systems, a proper sensor with some kind of focal plane pre-processing (such as spatial [2], [9], [10], [15], [16] or temporal [7]–[11] contrast, or motion extraction [12], [13]) is used. Nevertheless, in this section we show a luminance rate-coding experiment to illustrate "*pseudo-simultaneity*" and "*first-flash feature-extraction*" [43].

Fig. 13 illustrates event flow processing for a luminance rate-coded input event burst. The ConvModule was loaded with a vertical 11×7 Gabor filter to extract vertical edges. Input flow codes pixel light intensity with number of events per pixel (rate coding). In Fig. 13(a) each pixel uses 0 to 16 events. Visual input used 94×94 pixels and needed 90 k events. An 11 line kernel needs 260 ns to be processed ($4 + 22$ clock cycles at 100 MHz). However, only events for the central 74×70 pixels will be processed, as kernel size is 11×7 and ConvModule pixel array is 64×64. These 90 k events were processed in 14.9 ms, and Fig. 13(b) shows a representation of the 51 k output events produced by the ConvModule during the same 14.9 ms. In Fig. 13(c) and (d) we show the reconstructed input and output images when collecting the events during the first 3.2 ms only, and Fig. 13(e) and (f) corresponds to collecting events during the first 1.7 ms only. As can be seen, reconstructed input and output images degrade as less events per pixel are available. However, vertical edges are detected quite well even during the first milliseconds (see Fig. 13(d)), so that a fast preliminary feature extraction ("*first-flash feature-extraction*") takes place during the first events of the input flow. Fig. 13 illustrates nicely the "*pseudo-simultaneity*" property between input and output flows of Event-Driven Convolutional processing, as input and output events belong to the same time interval.

C. High Speed Moving Stimuli. Rotating Propellers

Fig. 13 illustrates Event-Driven convolution operation on static input visual flow. Dynamic Vision Sensors are meant for dynamic inputs, specially for very high speed, as each sensor pixel computes the temporal derivative of the sensed

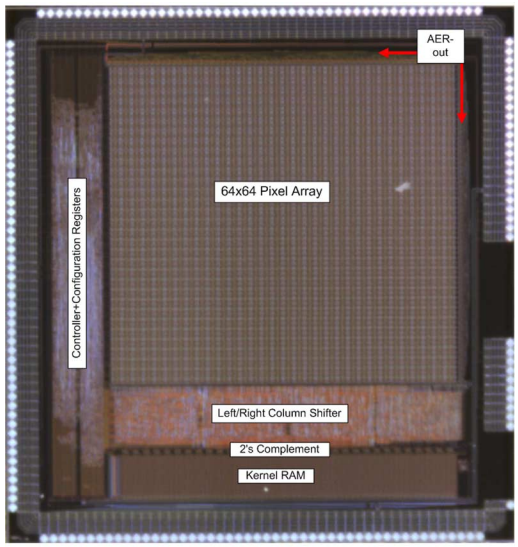

Fig. 10. Chip microphotograph.

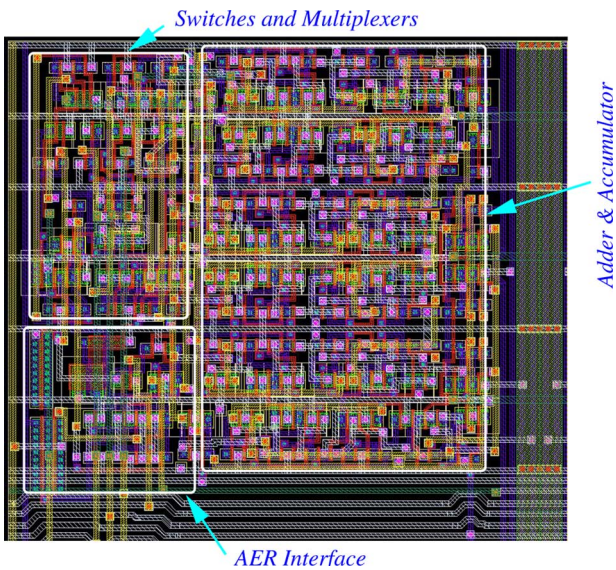

Fig. 11. Close-up of pixel layout indicating main parts.

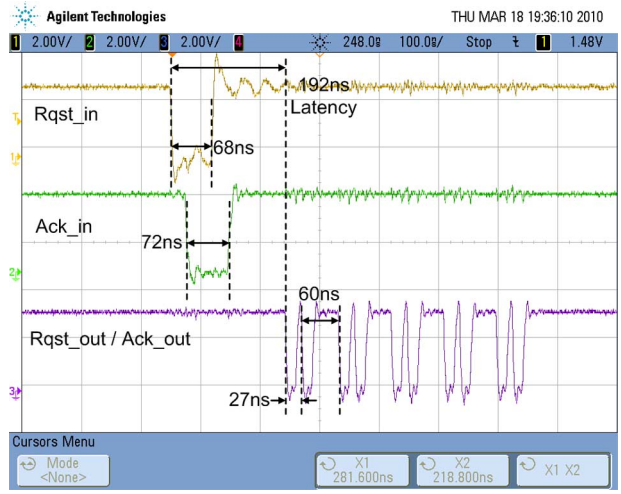

Fig. 12. Timing characterizations of input and output handshaking signals.

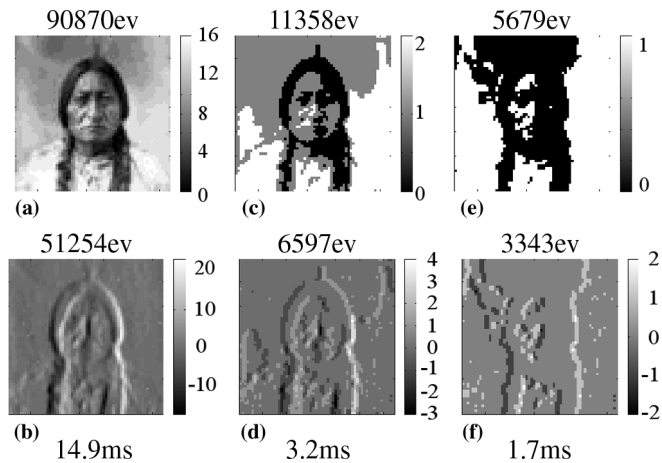

Fig. 13. Illustration of Event-Driven Luminance coding and convolution processing. Kernel is an 11×7 vertical Gabor filter. Input space is 94×94 pixels and output space is 64×64 pixels. (a) Full reconstruction of the 14.9 ms 90 k event burst representing a luminance image with pixels having between 0 to 16 events each. (b) Convolution output 51 k event burst produced during the same 14.9 ms, with pixels producing between 0 to 20 signed events each. (c) Input reconstruction of the first 11 k input events of the first 3.2 ms, with pixels having between 0 to 2 events each. (d) Output reconstructed from the first 6.6 k events of the first 3.2 ms, with pixels producing between 0 to 4 signed events each. (e) Input reconstruction of the first 5.6 k input events of the first 1.7 ms, with pixels having between 0 to 1 events each. (f) Output reconstructed from the first 3.3 k events of the first 1.7 ms, with pixels producing between 0 to 2 signed events each.

light. Fig. 14 illustrates the operation of the ConvModule when receiving the events of two propellers of different shapes rotating at 2000 revolutions per second, while moving slowly across the field of view. These events are generated artificially in a computer following the method explained elsewhere [40] and sent to the ConvModule chip using an event data player [44]. Fig. 14(a) shows the events collected during 50 ms (100 revolutions), while Fig. 14(b) for 50 μs (one tenth of a revolution). The ConvModule was programmed with a kernel to detect the S-shaped propeller in horizontal position by simple template matching: positive weights for propeller pixels and negative weights for surrounding pixels. Kernel size is 23×23. Fig. 14(c) shows the kernel weights, and Fig. 14(d) the output of the ConvModule where the events follow the center of the S-shape propeller and ignore the rectilinear one.

D. Combining the ConvModule With a DVS Retina

The high-speed event based processing example in Fig. 14 uses synthetic input data. Fig. 15 illustrates Event-Driven pattern recognition processing using real data from a 128×128 pixel Event-Driven DVS retina [8]. Performing robust shape and scale invariant pattern recognition with Convolutional Neural Networks requires large number of convolution modules, which is beyond the scope of the present article. However, we can illustrate the high-speed potential for pattern recognition by simple template matching. This is the case illustrated in Fig. 15. The retina events are filtered by a first center-on ConvModule and the output events are then processed by a second ConvModule for template matching. Thus, our recognition

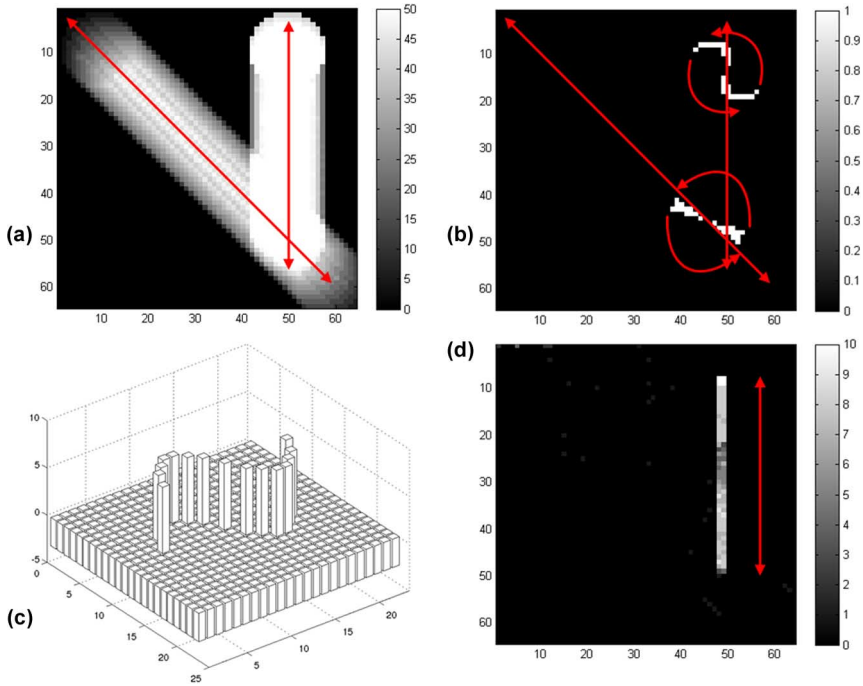

Fig. 14. 2 k rps propeller recognition. (a) 50 ms time capture of input stimulus events (100 revolutions). (b) 50 μs time capture of input stimulus events (1/10 revolution). (c) Output events produced by the Event-Driven ConvModule during 50 ms, following the center of the S-shaped propeller. (d) Kernel programmed.

system is a simple sequence of two feed forward ConvModule processing (see Fig. 15(a)). The retina is observing a person browsing a 52 card deck at high speed, as shown in Fig. 15(b). The 52 cards are swept in about 650 ms, although most of them are browsed in 410 ms (see Fig. 15(c)), which gives an average rate of about 8 ms per card. During these 410 ms the DVS retina [8] generates about 460 k events, while peak event rate goes up to 8 Meps. The DVS retina produces signed events. The sign bit is '1' for ON events (pixel light changed from dark to bright) or '0' for OFF events (pixel light changed from bright to dark). We filtered out the OFF events and sent only the ON events to the first ConvModule, programmed with the 7×7 center-on kernel shown in Fig. 16(b), to enhance edges and symbols and filter out noise. The ConvModule can "see" the full 128×128 input pixel space, but produces output events only for a 64×64 pixel window. Fig. 15(d) shows a 5 ms event capture from the ON events sent to the first ConvModule obtained with the jAER tool [46]. The output events of this first ConvModule are fed to the second ConvModule programmed with the 31×31 pixel template matching kernel shown in Fig. 15(e), to detect the "clover" symbols on the cards. The second ConvModule also produces 64×64 pixel output events, but we discard the most external 15 pixel ring to avoid false detections by undesired edge effects. Fig. 15(f) shows a y versus *time* projection of the events captured during 85 ms at the retina output (small dots), first ConvModule output (circles) and second ConvModule output (crosses). During these 85 ms six cards are browsed. Cards 2 to 4 contain "clover" symbols, which are correctly detected by the second ConvModule. Fig. 15(f) includes a 3 ms zoom box at "card 2", which are the (x, y, t) events for one "clover" symbol. The events within this box are zoomed out in Fig. 15(g) as y versus *time* and in Fig. 15(h) as y versus x. As can be seen, the retina events for this "clover" symbol start at about 26.3 ms and end at about 29 ms (2.7 ms total event time). The first convolution output events start at 26.6 ms (300 μs after the input first event) and end at about 28 ms (1.4 ms total event time). And the output events of the second convolution come out at 27.8 ms, 1.5 ms after the first retina event and 1.2 ms before the last retina event, or 1.2 ms after the first filter output event and 0.2 ms before the last one. Consequently, this example also illustrates nicely the pseudo-simultaneity processing of event-based systems, as recognition is achieved while the input stimulus events are being produced by the retina.

A final experiment that illustrates real data high-speed processing using the multi-kernel capability of the ConvModule is shown in Fig. 16(a). The same DVS retina [8] is looking at a 5 KHz spiral on an oscilloscope (as in Fig. 1). The retina sends its 128×128 pixel output events (x, y, s, k) to a merger module [44], where (x, y) are the retina pixel coordinates, 's' is event sign, and 'k' is a hard-wired 5-bit code to indicate the origin module of the event. Thus all events produced by the retina always have the same value 'k'. The merger output goes to the ConvModule, whose output events (x, y, s, k) are fed back to the second merger input. The ConvModule "sees" all 128×128 pixels but produces events only for a 64×64 pixel central window. The events produced by the ConvModule have a different code 'k' than the retina. Consequently, the ConvModule receives input events from the retina and from itself and will apply a different kernel depending on the origin of the events. For the retina events it uses the center-on kernel in Fig. 16(b) to enhance the center of the oscilloscope trace. And for the events coming out of itself it uses the kernel in Fig. 16(c). This kernel excites close by neighbors while it inhibits more dis-

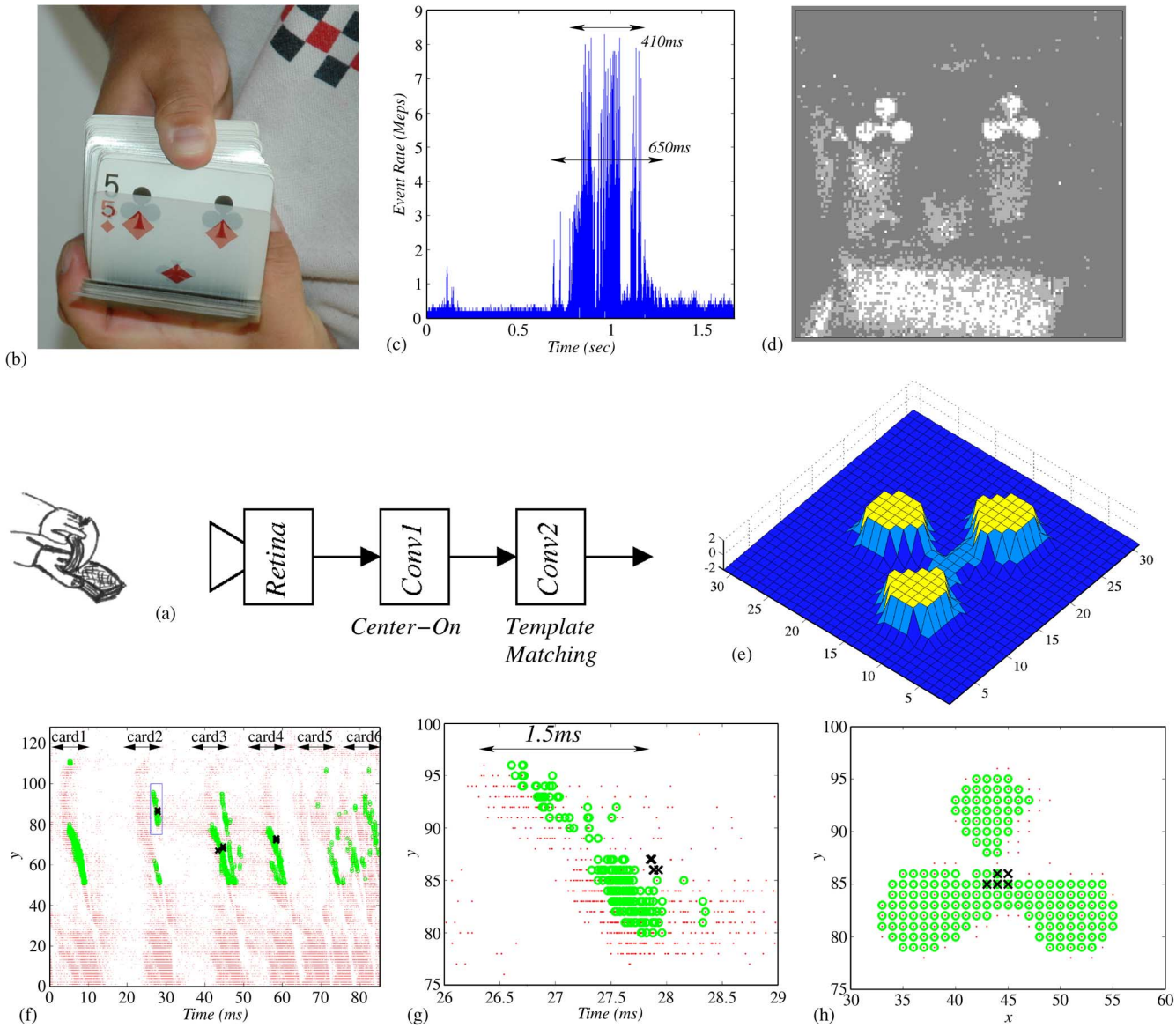

Fig. 15. Browsing a 52 card deck. (a) Setup of experiment. (b) Snapshot of 1/60 exposure (17 ms) taken with a commercial camera. (c) Instantaneous event rate of the half million events captured during 1.7 sec. (d) 5 ms event capture using a temporal derivative Event-Driven DVS retina. (e) 31 × 31 kernel used to detect clover symbols. (f) 85 ms event capture from the retina output (small dots), first convolution output (circles) and second convolution output (crosses). Events are projected on the y-time axes. During these 85 ms 6 cards are browsed. Cards 2 to 4 contain clover symbols. In 'card2' a 3 ms zoom box is shown. The events corresponding to this zoom box are shown in (g) as y versus *time* and in (h) as y versus x.

tant neighboring pixels, thus performing a soft-winner-takes-all competition among ConvModule pixels. The result of this processing is a sequence of small number of events following the center of the oscilloscope trace. Fig. 16(d) shows in 3D coordinates (x, y, t) the 825 events from the retina (small dots) and the 90 events from the output of the ConvModule (thick dots with line) during 500 μs. The trace center is followed with a delay of about 10 μs with respect to the center of the input data events.

A similar experiment was reported in the past [44] using ConvModule chips and a dedicated Event-Driven Winner-Takes-All (WTA) chip to detect and follow the center of a rotating circle. However, the maximum rotating speed reported then was 4 revolutions per second and the WTA output had a delay with respect to the retina first stimuli events of about 4 ms.

V. Scalability and Future Projections

The availability of multi-kernel event-driven ConvModules allows the assembly of arbitrary scale event-driven Convolutional Neural Networks (ConvNets). Here we try to estimate a futuristic projected performance of such systems.

The circuit techniques presented in this paper for building a 64×64 pixel ConvModule in a 5.5×5.8 mm^2 die in 0.35 μm can be extrapolated for modern 40 nm technologies. Pixel dimensions could be expected to be reduced by a factor of about 10. Consequently, increasing die area to about 1 cm^2, would make it realistic to handle pixel arrays of over one mega pixel, while emulating about one giga synapses (with projection fields of size 32×32). Handling such large arrays results also in longer event words, which would benefit from some kind of "event compression" technique to optimize communication bandwidth [47]. In

order to improve event throughput, processing pixels should be tiled into slices to avoid very long lines and pipeline/parallelize event processing. Off-chip event communication should be done serially [48], [49], and possibly using multiple I/O ports to improve inter-chip throughput. All this could probably improve event throughput by a factor of 100 with respect to the presented prototype. Consequently, we might consider as viable, event throughputs in the order of 10^8–10^9 eps (events per second) per chip.

Then, many of these chips (in the order of a hundred) could be assembled on a single PCB in a 2D grid array [50]. One such (stackable) PCB could emulate a neural ConvNet structure holding in the order of 10^8 neurons with 10^{11} synapses, which is about 1% of the human brain.

However, the most interesting way of scaling (besides scaling pixel array size) is to allow for multiple size-configurable ConvModules with configurable inter-module AER links within a single NoC (network on chip) die. This way, a single mega pixel (or mega neuron) size 40 nm NoC die could hold large numbers of individual ConvModules capable of interchanging intra-chip events at extremely high speeds. Note that ConvNets have usually large pixel array ConvModules in the first layers, but then their size reduces in subsequent layers, while the number of ConvModules is maximum at intermediate layers. For example, a standard face recognition ConvNet for 640×480 pixel inputs [32] uses four sequential layers: the first layer with 2.4×10^6 neurons (in 8 ConvModules each with one 5×5 kernel), the second with 1.5×10^6 neurons (in 20 ConvModules, each with 8 5×5 kernels), the third with 3.5×10^5 neurons (in 20 ConvModules, each with 20 5×5 kernels), and the fourth with 1.6×10^5 neurons (in 9 ConvModules, each with 180 1×1 kernels). Thus, total number of neurons is about 4.4×10^6, which could be fit into about five 40 nm 1 cm^2 NoC dies.

Regarding size of kernel-RAM, reported ConvNet systems for object recognition type tasks use kernel sizes in the order of 10×10 at the most, for VGA-like input images. On the other hand, what is more relevant for scaling up ConvNets is to have room for enough kernels per ConvModule. The maximum number of kernels per ConvModule is usually given by the number of ConvModules in the previous layer, which is normally less than a hundred. Therefore, when scaling up, each ConvModule should allow for about 100 10×10 kernels, at least, or 10^4 kernel values.

Regarding speed performance, since event-driven processing presents the *pseudo-simultaneity* property, recognition would be performed as soon as a sensor provides enough representative input events. A 128×128 pixel DVS sensor observing moving objects (as in Fig. 3) generates 10–100 keps, and a time between 10–40 ms provides enough events to recognize objects moving at normal life speeds (the high-speed card symbols in Fig. 15 needed about 10 times less time). Shaking the DVS sensor to observe static objects increases event rate to about 1–2 Meps. A 512×512 pixel DVS sensor would generate 16 times more events, but then objects can be recognized in less time, as enough representative events would be available about 16 times earlier. Consequently, we estimate that a reasonable recognition time for an event-driven ConvNet fed by a 512×512 pixel sensor could be in the order of a millisecond (independently of its scale).

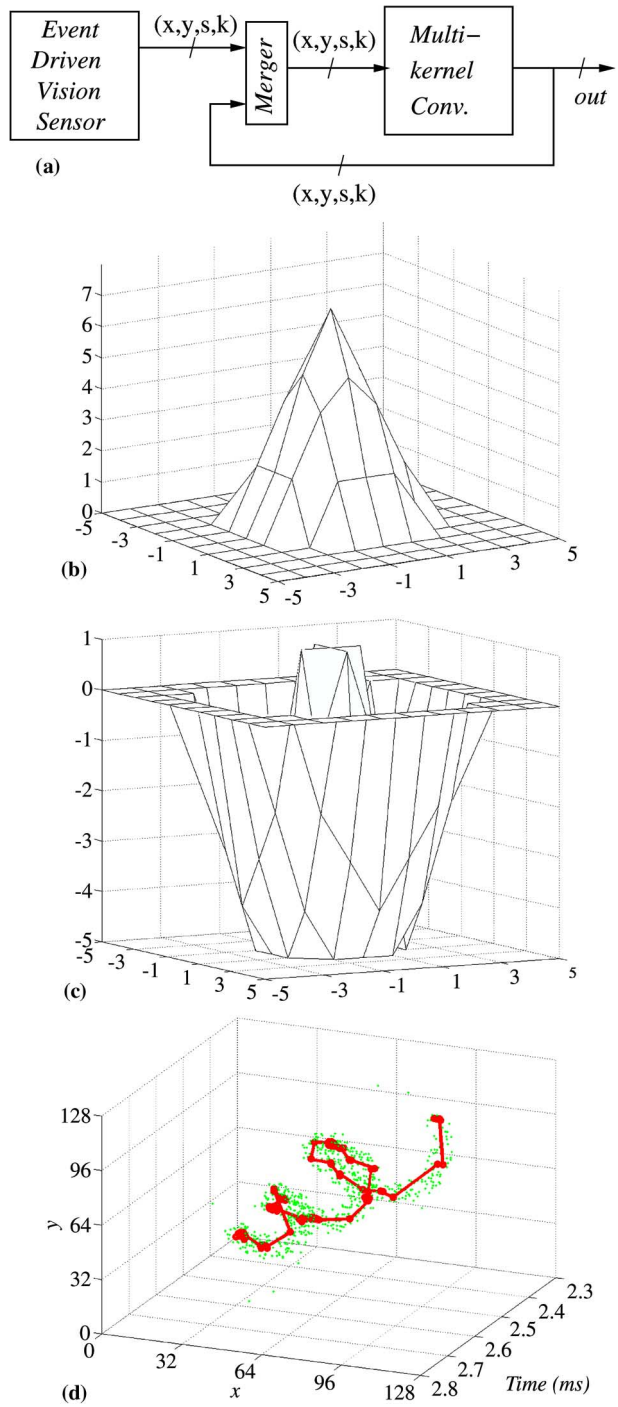

Fig. 16. Illustration of High-Speed Multi-kernel operation. (a) Setup: An Event-Driven Vision Sensor is observing a 5 KHz spiral on a phosphor oscilloscope (see Fig. 1) and is sending its output events to the Convolution Chip through an event merger module; the convolution module output is fed back to its input through the merger module. The ConvModule selects kernel (b) if the event comes from the vision sensor, or kernel (c) if it comes from its own output. Kernel (b) intends to highlight a circular shape of about 5 pixel diameter, while kernel (c) implements a soft winner-takes-all type of inter-pixel competition through mutual inhibition and slight self-amplification. (d) The result is that the ConvModule output events (large circles) follow closely the center of the oscilloscope trace (small dots) with an average update time of about 10 μs.

TABLE III
FRAME-BASED CONVNETS FOR FACE DETECTION

	Nasse [33]	NEC [32]	Yale/NYU [34]-[35]
hardware	GPU Nvidia GeForce 8800GT	FPGA Virtex-5	FPGA Virtex-6
input image size	640x480	640x480	512x512
total neurons	$4.9\text{x}10^6$	$4.4\text{x}10^6$	$2.4\text{x}10^6$
total synapses	$930\text{x}10^6$	$530\text{x}10^6$	$115\text{x}10^6$
peak MAC/s	N/A	$3.37\text{x}10^9$	$160\text{x}10^9$
delay	209ms	160ms	6ms
connections/s	$4.5\text{x}10^9$	$3.3\text{x}10^9$	$19.2\text{x}10^9$

VI. COMPARISON WITH STATE-OF-THE-ART FRAME-BASED GPU AND FPGA CONVNETS

Some researchers have reported GPU and FPGA-based hardware realizations of sophisticated Frame-based ConvNets for recognition type of applications, showing extraordinary performance figures, for both recognition rates and processing times. Table III summarizes recognition delays of three example systems that perform face recognition with ConvNets, implemented either with GPUs [33] or FPGAs [30], [32], [34], [35], when using VGA-like size input images. Nasse's implementation on an Nvidia GeForce 8800GT GPU needs 209 ms per input frame. NEC's system implemented on a Virtex-5 FPGA requires 160 ms, while Yale/NYU's system on a recent Virtex-6 FPGA can do a similar task in 6 ms.

The main advantage of Frame-based realizations is that the hardware can be time-multiplexed by fetching intermediate data between the processor and external memory, at the cost of slowing down speed performance. Time-multiplexing is not possible with Event-driven hardware as each neuron holds its state at each instant. On the other hand, one main advantage of event-driven hardware is that, because of the *pseudo-simultaneity* property, processing delay is kept approximately constant as hardware scales up. Another advantage is that up-scaling is simple by simply assembling more modules through asynchronous interconnect AER buses. However, large scale event-driven ConvNets are still under development.

VII. CONCLUSIONS

We have presented an Event-Driven Multi-Kernel Convolution Module chip for performing 2D kernel convolutions on visual data coming from Event-Driven Dynamic Vision Sensors (DVS). A ConvModule proof-of-concept test prototype has been fabricated in a 0.35 μm CMOS process, occupies a small area of $5.5\times5.8\,\text{mm}^2$, and can process an array of 64×64 pixels. Consequently, it is quite realistic to integrate several tens of such modules in a Network on Chip (NoC) die fabricated in a modern sub-100-nm CMOS technology. The presented ConvModule includes multi-kernel capability, which is a key property for assembling modular hierarchical multi-ConvModule systems for object recognition systems, following the computational paradigm known as "Convolutional Neural Networks". Event-driven sensing and processing systems turn out to present very high-speed visual processing and recognition capability, as events produced by the sensors are processed immediately by subsequent stages, event by event, resulting in "*pseudo-simultaneity*" between input and output visual information event streams. We have illustrated this by a series of experiments on static and dynamic visual event-driven data. The presented ConvModule could discriminate between propellers of different shapes rotating at speeds of up to 2000 revolutions per second, it could detect playing card symbols when browsing cards at an average rate of 8 ms per card and producing recognition events while the card is still being displayed, or it could detect and follow the center of an oscilloscope trace rotating at 5 KHz.

Future work will concentrate on assembly and reconfigurability strategies for assembling tens or hundreds of Event-Driven ConvModules for generic object recognition tasks.

ACKNOWLEDGMENT

The authors are grateful to Tobi Delbrück for valuable discussions, providing an initial AER temporal contrast retina [7] and the jAER open software [46], Anton Civit's group for the AER interfacing and data record/playback boards [44], Philipp Häfliger for the CAVIAR PCB for holding the ConvChip and the lens mount holder for the retina, and Eugenio Cullurciello and Clement Farabert for valuable discussions. The authors are also grateful to Simon Thorpe for valuable discussions on event based processing and suggesting the idea of implementing the multi-kernel capability.

REFERENCES

[1] E. Culurciello, R. Etienne-Cummings, and K. A. Boahen, "A biomorphic digital image sensor," *IEEE J. Solid-State Circuits*, vol. 38, pp. 281–294, 2003.

[2] P. F. Ruedi *et al.*, "A 128 × 128 pixel 120-dB dynamic-range vision sensor chip for image contrast and orientation extraction," *IEEE J. Solid-State Circuits*, vol. 38, pp. 2325–2333, 2003.

[3] M. Barbaro, P. Y. Burgi, A. Mortara, P. Nussbaum, and F. Heitger, "A 100 × 100 pixel silicon retina for gradient extraction with steering filter capabilities and temporal output coding," *IEEE J. Solid-State Circuits*, vol. 37, pp. 160–172, 2002.

[4] C. Shunshun and A. Bermak, "Arbitrated time-to-first spike CMOS image sensor with on-chip histogram equalization," *IEEE Trans. VLSI Syst.*, vol. 15, no. 3, pp. 346–357, Mar. 2007.

[5] M. Azadmehr, H. Abrahamsen, and P. Hafliger, "A foveated AER imager chip," presented at the IEEE Int. Symp. Circuits and Syst., Kobe, Japan, 2005.

[6] R. J. Vogelstien, U. Mallik, E. Culurciello, R. Etienne-Cummings, and G. Cauwenberghs, "Spatial acuity modulation of an address-event imager," in *Proc. IEEE Int. Conf. Electronics, Circuits and Systems*, 2004, pp. 207–210.

[7] 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.

[8] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 3.6 μm latency asynchronous frame-free event-based dynamic vision sensor," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1443–1455, Jun. 2011.

[9] K. A. Zaghloul and K. Boahen, "Optic nerve signals in a neuromorphic chip I: Outer and inner retina models," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 4, pp. 657–666, Apr. 2004.

[10] K. A. Zaghloul and K. Boahen, "Optic nerve signals in a neuromorphic chip II: Testing and results," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 4, pp. 667–675, Apr. 2004.

[11] 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.

[12] 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.

[13] E. Ozalevli and C. M. Higgins, "Reconfigurable biologically inspired visual motion system using modular neuromorphic VLSI chips," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 1, pp. 79–92, 2005.

[14] K. Boahen and A. Andreou, "A contrast-sensitive retina with reciprocal synapses," *Advances in Neural Information Processing Systems (NIPS)*, vol. 4, pp. 764–772, 1992.

[15] 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 Syst. I*, vol. 54, no. 7, pp. 1444–58, 2007.

[16] 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 0.1 ms latency and optional time-to-first-spike mode," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 10, pp. 2632–2643, Oct. 2010.

[17] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *J. Physiol.*, no. 148, pp. 574–591, 1959.

[18] E. T. Rolls and G. Deco, *Computational Neuroscience of Vision*. Oxford, U.K.: Oxford University Press, 2002.

[19] 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.

[20] T. Masquelier and S. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Comp. Biol.*, vol. 3, no. 2, p. e31, 2007, doi 10.1371/journal.pcbi.0030031.

[21] K. Fukushima, "Visual feature extraction by a multilayered network of analog threshold elements," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-5, no. 4, pp. 322–333, Oct. 1969.

[22] B. E. Boser, E. Säckinger, J. Bromley, Y. LeCun, and L. D. Jackel, "An analog neural network processor with programmable topology," *IEEE J. Solid-State Circuits*, vol. 26, no. 12, pp. 2017–2025, Dec. 1991.

[23] 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 Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[24] K. Chellapilla, M. Shilman, and P. Simard, "Optimally combining a cascade of classifiers," in *Proc. Document Recognition and Retrieval*, Jan. 2006, vol. 13, Electronic Imaging, 6067.

[25] 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.

[26] M. Osadchy, Y. LeCun, and M. Miller, "Synergistic face detection and pose estimation with energy-based models," *J. Mach. Learn. Res.*, vol. 8, pp. 1197–1215, May 2007.

[27] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1408–1423, 2004.

[28] F. Nasse, C. Thurau, and G. A. Fink, "Face detection using gpu based convolutional neural networks," *Computer Analysis of Images and Patterns*, vol. 5702/2009, Lecture Notes in Computer Science, pp. 83–90, 2009.

[29] 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," in *Int. Conf. Comput. Vision (ICCV'09)*, Kyoto, Japan, 2009, pp. 2373–2380.

[30] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "CNP: An FPGA-based processor for convolutional networks," in *Proc. Int. Conf. Field Programmable Logic and Applications*, Prague, Czech Republic, 2009, pp. 32–37.

[31] C. Farabet, B. Martini, P. Akserod, S. Talay, Y. LeCun, and E. culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, Paris, France, 2010, pp. 257–260.

[32] M. Sankaradas, V. Jakkul, S. Cadambi, S. Chakradhar, I. Durdanovic, E. Cosatto, and H. P. Graf, "A massively parallel coprocessor for convolutional neural networks," in *Proc. 20th IEEE Int. Conf. Application-Specific Systems, Architecture, and Processing.*, 2009, pp. 53–58.

[33] F. Nasse, C. Thurau, and G. A. Fink, "Face detection using GPU-based convolutional neural networks," *Computer Analysis of Images and Patterns*, vol. 5702, Lecture Notes in Computer Science, pp. 83–90, 2009.

[34] C. Farabet, Y. LeCun, K. Kavukcuoglu, E. Culurciello, B. Martini, P. Akselrod, and S. Talay, "Large-scale FPGA-based convolutional networks," in *Machine Learning on Very Large Data Sets*, R. Bekkerman, M. Bilenko, and J. Langford, Eds. Cambridge, U.K.: Cambridge University Press, 2011.

[35] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "NeuFlow: A runtime-reconfigurable dataflow processor for vision," in *Proc. Embedded Computer Vision Workshop (ECVW'11)*, 2011 [Online]. Available: https://engineering.purdue.edu/elab/research/svision/svision.html

[36] M. A. Mahowald, "VLSI analogs of neuronal visual processing: A synthesis of form and function," Ph.D. dissertation, Computation and Neural Systems, Caltech, Pasadena, CA, 1992.

[37] T. Serrano-Gotarredona, A. G. Andreou, and B. Linares-Barranco, "AER image filtering architecture for vision processing systems," *IEEE Trans. Circuits Syst. I*, vol. 46, no. 9, pp. 1064–1071, Sep. 1999.

[38] P. Venier, A. Mortara, X. Arreguit, and E. A. Vittoz, "An integrated cortical layer for orientation enhancement," *IEEE J. Solid-State Circuits*, vol. 32, no. 2, pp. 177–186, Feb. 1997.

[39] T. Y. W. Choi, P. Merolla, J. Arthur, K. Boahen, and B. E. Shi, "Neuromorphic implementation of orientation hypercolumns," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 6, pp. 1049–1060, Jun. 2005.

[40] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jiménez, and B. Linares-Barranco, "A neuromorphic cortical-layer microchip for spike-based event processing vision systems," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 53, no. 12, pp. 2548–2566, Dec. 2006.

[41] L. Camuñas-Mesa, A. Acosta-Jiménez, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 32 $\times$ 32 pixel convolution processor chip for address event vision sensors with 155 ns event latency and 20 Meps throughput," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 4, pp. 777–790, Apr. 2011.

[42] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits Syst. II*, vol. 47, no. 5, pp. 416–434, May 2000.

[43] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *Nature*, vol. 381, pp. 520–522, Jun. 6, 1996.

[44] 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 45 k-neuron, 5 M-synapse, 12 G-connects/sec AER hardware sensory-processing-learning-actuating system for high speed visual object recognition and tracking," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1417–1438, Sep. 2009.

[45] *Microscopic Particle Tracking Using a DVS Retina*. Inst. Neuroinformatics, Zurich, Switzerland. [Online]. Available: http://siliconretina.ini.uzh.ch

[46] *jAER Open Source Project*. Inst. Neuroinformatics, Zurich, Switzerland. [Online]. Available: http://jaer.wiki.sourceforge.net

[47] D. G. Chen, A. Bermak, and C. Y. Tsui, "A low-complexity image compression algorithm for Address-Event Representation (AER) PWM image sensors," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2010)*, Rio de Janeiro, Brazil, 2010, pp. 2825–2828.

[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 Syst. I*, in press, doi: 10.1109/TCSI.2011.2151070.

[49] C. Zamarreño-Ramos, T. Serrano-Gotarredona, B. Linares-Barranco, R. Kulkarni, and J. Silva-Martinez, "Voltage mode driver for low power transmission of high speed serial AER links," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2011)*, Rio de Janeiro, Brazil, May 15–18, 2011, pp. 2433–2436.

[50] M. Khan, D. Lester, L. Plana, A. Rast, X. Jin, E. Painkras, and S. Furber, "Spinnaker: Mapping neural networks onto a massively-parallel chip multiprocessor," in *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN)*, Jun. 2008, pp. 2849–2856.