A Neuromorphic Cortical-Layer Microchip for Spike-Based Event Processing Vision Systems

Rafael Serrano-Gotarredona, Teresa Serrano-Gotarredona, Antonio Acosta-Jiménez, and Bernabé Linares-Barranco

***Abstract*—We present a neuromorphic cortical-layer processing microchip for address event representation (AER) spike-based processing systems. The microchip computes 2-D convolutions of video information represented in AER format in real time. AER, as opposed to conventional frame-based video representation, describes visual information as a sequence of events or spikes in a way similar to biological brains. This format allows for fast information identification and processing, without waiting to process complete image frames. The neuromorphic cortical-layer processing microchip presented in this paper computes convolutions of programmable kernels over the AER visual input information flow. It not only computes convolutions but also allows for a programmable forgetting rate, which in turn allows for a bio-inspired coincidence detection processing. Kernels are programmable and can be of arbitrary shape and arbitrary size of up to 32 × 32 pixels. The convolution processor operates on a pixel array of size 32 × 32, but can process an input space of up to 128 × 128 pixels. Larger pixel arrays can be directly processed by tiling arrays of chips. The chip receives and generates data in AER format, which is asynchronous and digital. However, its internal operation is based on analog low-current circuit techniques. The paper describes the architecture of the chip and circuits used for the pixels, including calibration techniques to overcome mismatch. Extensive experimental results are provided, describing pixel operation and calibration, convolution processing with and without forgetting, and high-speed recognition experiments like discriminating rotating propellers of different shape rotating at speeds of up to 5000 revolutions per second.**

***Index Terms*—2-D convolutions, address-event representation (AER), bio-inspired systems, digitally calibrated analog circuits, high-speed signal processing, MOS transistor mismatch, spike-based processing, subthreshold circuits, vision, VLSI mixed-circuit design.**

I. Introduction

ARTIFICIAL man-made machine vision systems operate in a quite different way than do biological brains. Machine vision systems usually operate by capturing and processing sequences of frames. For example, a video camera captures images at about 25–30 frames per second, which are then processed frame by frame to extract, enhance, and combine features and perform operations in feature spaces until a desired recognition is achieved. Biological brains do not operate on a frame-by-frame basis. In the retina, each pixel sends spikes (also called events) to the cortex when its activity level reaches a threshold. This activity level may respond to different image properties like intensity, contrast, color, and motion—properties that have been precomputed within the retina before generating the spikes to be sent to the visual cortex. Very active pixels will send more spikes than less active pixels. When the retina responds to a stimulus, for example, a moving profile, then those pixels sensing the profile will elicit a simultaneous collection of spikes that are strongly space-time correlated. The visual cortex receiving these spikes is sensitive to the space location where the spikes were originated and to the relative timing between them. This way, it can recognize and follow this moving profile. All of these spikes are transmitted as they are being produced and do not wait for an artificial "frame time" before sending them to the next processing layer. This way, in biological brains, strong features are propagated and processed from layer to layer as soon as they are produced, without waiting to finish collecting and processing data of whole image frames.

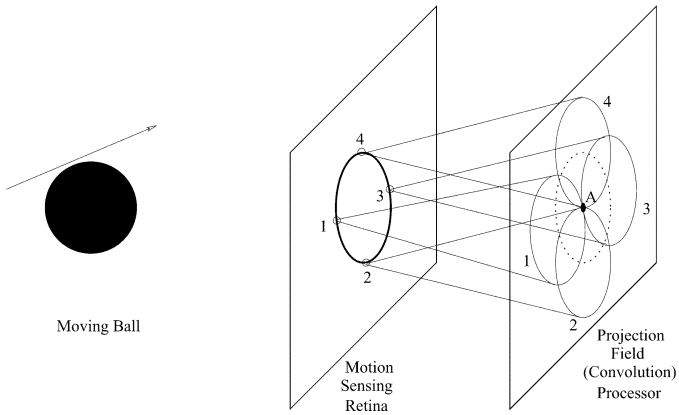


Fig. 1. Example of high-speed projection-field spike-based image processing for detecting a moving ball of a specific radius.

As an illustration, consider the setup in Fig. 1. On the left, a circular solid object (a ball) is observed by a motion-sensing retina in the center. The pixels in this retina are sensitive to motion (changes in intensity). Consequently, at a given instant in time, only the pixels on a circumference will become active. This means that the pixels on the same circumference will simultaneously fire spikes. Let us assume that each pixel fires just one single spike. We may state that, at a given instant (or short time interval), the spikes produced by the retina are highly space-time correlated: in time because they are simultaneous and in space because they form a circumference of a certain radius. In Fig. 1, the output spikes of the retina are sent, through projection fields, onto the next processing layer. Suppose that

Manuscript received April 5, 2006. This work was supported by Spanish Research under Grant TIC2003-08164-C03-01 (SAMANTA) and by the European Union under EU Grant IST-2001-34124 (CAVIAR). The work of R. Serrano-Gotarredona was supported by the Spanish Ministry of Education and Science. This paper was recommended by Associate Editor A. van Schaik.

The authors are with the Instituto de Microelectrónica de Sevilla (IMSE), Centro Nacional de Microelectrónica (CNM), Consejo Superior de Investigaciones Cientificas (CSIC) and Universidad de Sevilla, 41012 Sevilla, Spain (e-mail: bernabe@imse.cnm.es).

Digital Object Identifier 10.1109/TCSI.2006.883843

the projection fields are tuned to detect circumferences of a given radius range $R\pm\varepsilon$. Then, each spike produced by a pixel in the retina will be sent to a circumference (of radius R) of pixels in the projection-field layer in Fig. 1. This way, pixel "1" in the retina sends a spike to all pixels in circumference "1" of the projection-field layer. The same is true for pixels "2," "3," "4," and all others in the retina circumference. If the circumference sensed in the retina is of the same radius R than the projection fields, as is the case in Fig. 1, then the pixel in the projection field layer that has the same coordinates as the central pixel of the retina circumference (pixel "A"), will receive spikes from all active projection fields. Consequently, this pixel will receive the strongest stimulus. The pixels in the projection-field layer can be made to fire a spike if their stimulus reaches a certain threshold. If this threshold is sufficiently high, only the central pixel "A" in the projection field layer will generate an output, signaling that this is the center of the moving ball of radius R sensed by the retina. In general, projection fields in biological neurocortical layers perform feature extraction operations, which are dependent on the "shape" (weights) of the projection-field connections. Note that projection-field processing is equivalent to convolution processing, where the kernel of the convolution is the projection-field shape. In the case of Fig. 1, the feature to be detected is a circumference of radius R. In biological neurocortical structures there are several (8–10) sequential projection-field layers that extract features [1]–[4], group them, extract more elaborate features, and so on, until in the end they perform complicated recognition tasks, such as handwritten character recognition [5]–[8] or face recognition [9]–[13].

A very interesting and powerful property of the projection-field processing illustrated in Fig. 1 is its high speed. Note that the spikes produced at the retina are sent simultaneously to the projection field layer. The central pixel A produces its output spike almost instantly. Consequently, this spike-based projection-field processing approach is structurally much faster than a conventional frame-based processing approach. In a frame-based approach, all pixels in a retina (or camera) are sensed and transmitted to the next layer (or processing stage), where all pixels of the frame are processed, usually with convolution operations, and so on. This frame convolution processing is slow, especially if several convolutions need to be computed in sequence for each input image frame.

The cortical layer microchip presented in this paper exploits the spiking projection-field computation principles of biological neurocortical structures, as illustrated in Fig. 1. For this microchip, the delay between consecutive layers is in the order of *nano* to *micro* seconds. Each pixel in the projection field layer includes an integrate-and-fire neuron (i&f) [20]–[23], [28]–[31], [42]–[47], [51], [56], [59]: the neuron integrates all incoming spikes and, when the integral reaches a threshold, the neuron fires its own output spike and resets itself. A circuit performing this conceptual operation is illustrated in Fig. 2(a). Every time a pixel receives a spike, a current pulse is integrated onto capacitor C, increasing its voltage V_c. When this voltage reaches threshold $E_{\rm ref}$, the capacitor is reset and an output spike is produced.

However, remember that, for the system in Fig. 1, we want to detect highly space-time-correlated features, like the moving

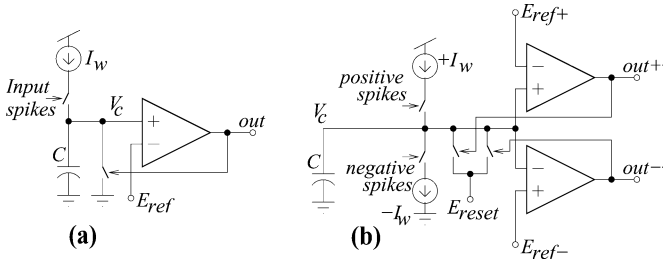


Fig. 2. Integrate-and-fire neuron conceptual circuit diagrams. (a) For unsigned processing. (b) For signed processing.

circumference. For example, the central pixel A in the projection-field layer detecting the center of the moving circumference has to detect this circumference if the incoming spikes are received within a very short time interval. Consequently, the i&f circuit of Fig. 2(a) should include some loss mechanism, so that, if the incoming spikes do not come in sufficiently fast, no output spike should be produced. Otherwise, two half circumferences separated in time could be recognized as a single circumference. This forgetting mechanism can be implemented by connecting a loss resistor in parallel with capacitor C or a leak-discharging current source. Whatever loss mechanism is used, it is essential for correct space-time-correlation feature detection. In the cortical-layer microchip presented in this paper, a precise loss mechanism has been implemented with programmable loss rate.

For generic convolution processing, signed input events have to be processed with signed convolution weights. Consequently, the current pulses to be integrated on capacitor C in Fig. 2 may be either positive or negative, as shown in Fig. 2(b) for a signed i&f neuron. Also, for signed integration, there will now be two limiting thresholds, a positive one $E_{\rm ref+}$ and a negative one $E_{\rm ref-}$. Depending on which one is reached, the i&f neuron will provide a positive event (visible at $out+$) or a negative one (visible at $out-$). Both events will reset the voltage of capacitor C to an intermediate reset value $E_{\rm reset} = (E_{\rm ref+} + E_{\rm ref-})/2$. For the signed i&f neuron of Fig. 2(b), a loss mechanism could be implemented by including a resistor between nodes V_c and $E_{\rm reset}$. However, since the required resistance value would be extremely large, this would not be practical. We will use a signed discharging current mechanism, which is explained in Section III.

The microchip described is based on a previously reported convolution chip, the address-event-representation (AER) processor [14], [76], which now includes a loss mechanism for proper cortical spike-based feature-detection operations.

AER is a spike-based representation hardware technique for communicating spikes between layers of neurons in different chips. AER was first proposed in 1991 in one of the Caltech research labs [15]–[19] and has been used since then by a wide community of neuromorphic hardware engineers. Unarbitrated and less sophisticated event read-out has been used [20], [21], and more elaborate and efficient arbitrated versions have also been proposed, based on Winner-Takes-All [22] or the use of arbiter trees [23], which have evolved to row parallel [24] and burst-mode word-serial [25]–[27] read-out schemes. AER has been used fundamentally in image sensors, for simple light intensity to frequency transformations [28], time-to-first-spike codings [29]–[32], foveated sensors [33],[55], [77], [78], and

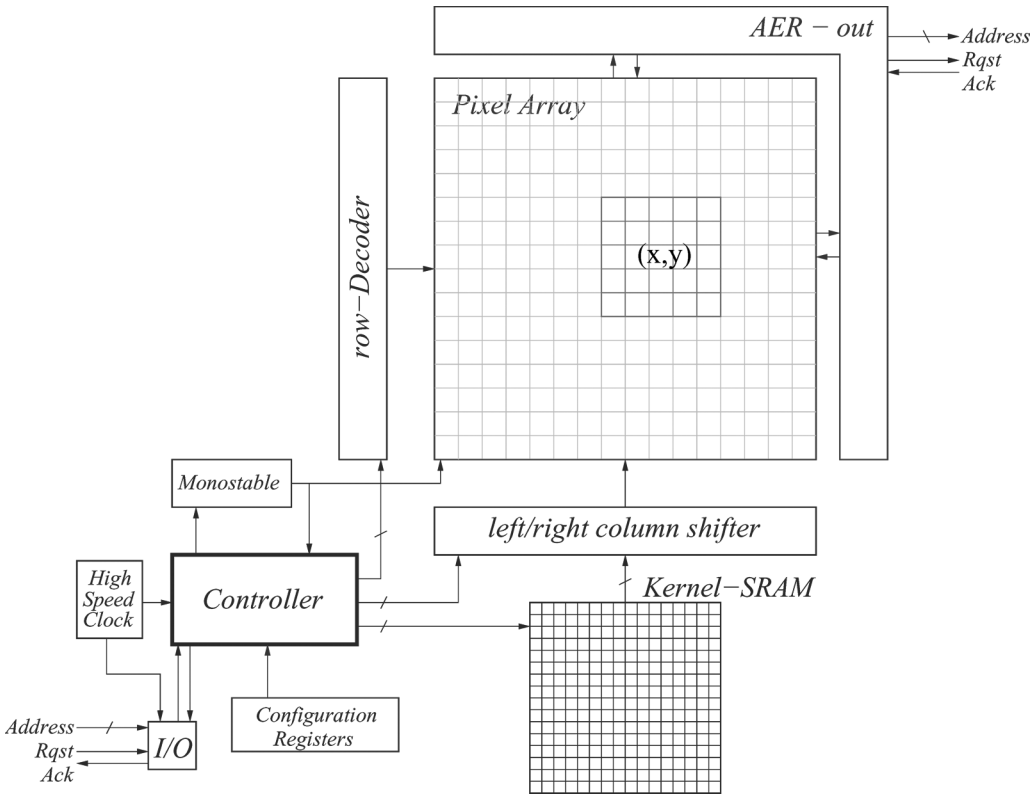


Fig. 3. Schematic architecture of convolution chip prototype.

more elaborate transient detectors [34]–[36] and motion sensing and computation systems [37]–[41]. However, AER has also been used for auditory systems [18], [42]–[44], competition and winner-takes-all networks [45]–[47], and even for systems distributed over wireless networks [48].

In this paper, we are concerned with the exploitation of the AER benefits for higher level artificial vision based on the processing found in biological cortical cerebral structures, specifically those based on projection fields (or convolutional operators). In this respect, some prior work is available in literature. Some works focus on creating projection fields with inter-chip digital machines like micro-controllers with lookup tables (LUTs) [50]–[56]. This technique is quite versatile but results in slower systems, since for each spike generated at the emitter end a "bubble" of sequential events is generated for the receiver. Other researchers directly implement the projection field inside the receiver chip without slowing down the single spike inter-chip communication speed, at the cost of introducing severe constraints on the shape of the projection fields. For example, Vernier *et al.* [57] restrict this to diffusive elliptical kernels, Choi *et al.* [59], [60] to Gabor-type kernels, and Serrano *et al.* [58] to x/y decomposable kernels. The projection field (or convolution) processor chip presented in this paper is an improvement of a prior system [14], [76], which does not impose any restrictions either on the shape or size of the kernel, does not slow down the AER inter-chip communication (except for very large kernels), and allows precise leaky integration for high-speed feature-detection operations. Thanks to this absence of kernel shape restrictions, this chip can be used as a module in complex hierarchical multilayer cortical-like vision architectures, like the BCS-FCS system [61]–[63], Fukushima's Neocognitron [5]–[8], or the Convolution Neural Networks [9]–[13] used for handwritten character recognition and face recognition. It has already been used in a sophisticated multilayer AER hierarchy for detecting moving balls [64].

This paper is structured as follows. In Section II, the architecture of the chip is briefly described. Section III provides design details of the circuitry of the i&f pixel with forgetting capability. Section IV shows extensive experimental results. Finally, the conclusions of the presented work are summarized in Section V.

II. Architecture of Programmable-Kernel AER Convolution Chip

The chip reported in this paper is an AER transceiver chip that performs arbitrary kernel convolutions. It receives input AER signals representing dynamic 2-D visual flow, computes convolutions on it, and generates the output as AER signals. The convolution processing is performed for each incoming event [14]: every time an input event is received, if (x, y) is its coordinate, spikes of modulated amplitudes will be sent to a projection field of pixels around this coordinate. The procedure can be explained with the help of Fig. 3, where the system-level architecture of the chip is illustrated. The chip contains the following elements:

- high-speed clock for the synchronous blocks;
- input I/O block for handling and receiving the incoming input events;
- synchronous controller for sequencing all operations to be performed for each incoming event;
- monostable that generates an integration pulse of fixed width;
- row decoder for selecting active rows;
- left/right column shift element;

- static RAM that holds the stored kernel;
- array of i&f neuron pixels;
- asynchronous circuitry for arbitrating and sending out the output address events generated by the array of i&f neurons;
- configuration register that stores configuration parameters loaded at startup, through a serial port.

The I/O block continuously scans the input AER port until an active Rqst is detected, in which case the corresponding event address (x, y) is latched and sent to the controller, and the asynchronous handshaking cycle is properly handled and completed. The controller[1] reads the input address, and, depending on the kernel size (stored in the configuration registers), it computes the limit coordinates of the projection field, around (x, y), onto which the convolution kernel stored in the static RAM needs to be copied. The output of this computation is: 1) a column left/right shift between the RAM columns holding the kernel and the projection field columns in the array and 2) the row address correspondence between kernel RAM rows and array rows. Note that, depending on the kernel size and event coordinate, the target projection field can fall fully inside the array (as illustrated in Fig. 3), partially overlap the array, or fall completely outside the array.[2] After this computation, the controller will copy, line after line, all kernel rows from the kernel RAM registers to registers in the pixels. These pixel registers will modulate a current pulse amplitude that will be integrated onto an integration capacitor in each pixel (I_w in Fig. 2). Once all of the pixel registers of the projection field are loaded with the kernel values, the controller triggers the monostable, which provides a global pulse of fixed width to all pixels in the array. When a pixel has integrated current pulses beyond its voltage threshold ($E_{\mathrm{ref}+}$ or $E_{\mathrm{ref}-}$ in Fig. 2), it will reset itself and generate an output event, which will be handled by the asynchronous AER circuitry and sent off chip with its corresponding handshaking signals.

The details of the peripheral circuits have been reported previously [14], [65], [76], and we will not repeat them here. The asynchronous AER circuit follows the row parallel event read-out technique developed by Boahen [24], except that it now handles signed events.

Handling signed events is important for computing convolutions. In general, convolution kernel values can be positive or negative. Therefore, the output values of convolutions can also have positive or negative values. Consequently, output address events must include a sign bit. Furthermore, since convolution operations can be cascaded, a generic AER convolution processor must be able to handle signed input events as well. In conclusion, the convolution chip processor described in this paper includes sign bits for the input and output address events and for the kernel values stored in the kernel RAM. Each integrator in the array of i&f neurons is therefore able to handle signed input events, signed output events, and signed kernel values and follows the scheme of Fig. 2(b).

[1]The controller operation was described algorithmically through vhdl code and synthesized automatically.

[2]The input address space the chip needs to process is larger than the address space of its own array of pixels: it is precisely its own address space expanded by the largest kernel size. For a chip with 32×32 pixels and 32×32 maximum kernel size, the input address space has to be at least $(32 + 2 \times 31)^2 = 93 \times 93$.

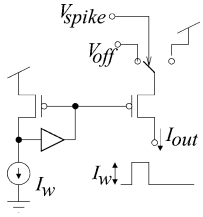


Fig. 4. Source pulsed current source.

Note that larger image pixel arrays can be processed by assembling 2-D arrays of convolution chips. Each chip would need to know its own address space within the full pixel space (set by configuration registers), and all chips would need to read in parallel the same input AER port [14], [76]. This chip tiling is possible because in the AER convolution operation (see Fig. 3) there is no interaction between neighboring pixels.

III. Calibrated i&f Neuron Pixel for Signed Event Processing With Forgetting Capability

Event convolution computations are performed by integrating signed and weighted current pulses in arrays of i&f neurons. Transistors are pulsed through their sources [50] to minimize charge injection and parasitic feedthrough charges. Fig. 4 shows the basic idea behind this scheme. The transistors form a current mirror. The input branch of the mirror is biased with a constant current I_w, and the gate is buffered with a simple source-follower circuit to minimize transients at the transistor gates. This mirror input is maintained at constant values all the time (no dynamics). The mirror output transistor switches its source between a voltage source of value V_{off} and the power supply. During a spike, the source is connected to the power supply and the mirror output provides a current of value I_w. This very simple scheme allows for very fast switching (pulses below 100-ns width) with very low currents (down to the picoamps range), while the charge packets delivered maintain very good linearity with bias current I_w over a large range [66]. Therefore, this circuit is very well suited for binary weighted current pulse amplitude modulation, down to the picoampere range. This is done in the following way.

Fig. 5 shows the schematic diagram of our i&f neuron with signed events and weight processing capability and forgetting functionality. Signals $\mathrm{bp}\langle 0 \ldots 2\rangle$ and $\mathrm{bn}\langle 0 \ldots 2\rangle$ control a set of switches, which by default connect the sources of transistors $M_{p1} \ldots M_{p3}$ and $M_{n1} \ldots M_{n3}$ to their off voltage (either V_{offp} or V_{offn}). The monostable at the periphery will send a pulse to all pixels in the array through either global line $Pulse+$ or $Pulse-$, depending on the sign of the incoming event. Depending on this sign and the sign of the convolution weight for this pixel (stored in a dynamic register inside the block labeled *Logic* in Fig. 5), either lines $\mathrm{bp}\langle 0 \ldots 2\rangle$ or $\mathrm{bn}\langle 0 \ldots 2\rangle$ will be activated during the time of this pulse and according to the value of the weight. The weights have a resolution of 4 bits, including a sign bit. Transistors M_{n1}–M_{n3} will provide binary weighted negatively signed pulses, while transistors to M_{p1}–M_{p3} will provide positively signed ones. Note that the sources of the mirror input and output branches are not connected directly to

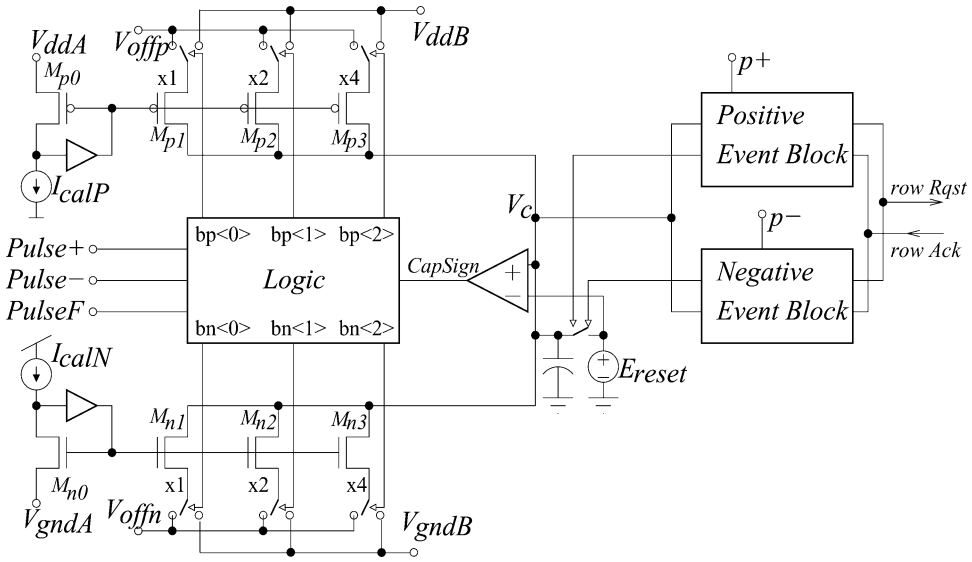


Fig. 5. Schematic diagram of signed i&f neuron with forgetting capability.

the power supply rails. There are two reasons for this: 1) by setting $V_{gndA} > V_{gndB}$ and $V_{ddA} < V_{ddB}$, a current gain will be introduced from the mirror inputs to their outputs, which allows reduction of supply consumption in standby and 2) separating the source voltages from the power rails allows for very low current processing [67], [68]. Transistors $M_{n/p1}$–$M_{n/p3}$ are binary weighted in size (each nMOS unit transistor is of size 1/4 μm and each pMOS one is of 2/6 μm). If the sign of the input event and stored weight are the same, lines $\mathrm{bp}\langle 0 \ldots 2\rangle$ will be activated (according to weight magnitude), otherwise lines $\mathrm{bn}\langle 0 \ldots 2\rangle$ will become active selectively.

The amplitude of the current pulses can be quite small. The integrating capacitor has a rather low value of 140 fF (20 μm $\times$ 8 μm in size) to reduce pixel area consumption. For our convolution chip prototype, maximum kernel size is 32 $\times$ 32, which means that a possible situation could arise in which a pixel has to fire when it receives pulses from 32 $\times$ 32 a neighborhood: it might be necessary to charge capacitor C after receiving $n_p = 32 \times 32 = 1024$ pulses. If the monostable pulsewidth is set to $T_h = 300$ ns and the integration voltage threshold to $\Delta V = E_{\mathrm{ref}-} - E_{\mathrm{reset}} = E_{\mathrm{reset}} - E_{\mathrm{ref}-} = 0.5$ V, then the current pulse values would be in the order of

$$I = \frac{C\Delta V}{n_p T_h} = \frac{140\ \text{fF} \times 0.5\ \text{V}}{1024 \times 300\ \text{ns}} \approx 230\ \text{pA} \tag{1}$$

and the pulse amplitude for the least significant bit would be a small fraction of this ($I/8 = 30$ pA). It is obvious that current pulsing transistors $M_{n/p1}$–$M_{n/p3}$ will be operated deep in the subthreshold region. Transistor mismatch under such conditions is very high [69]. Fig. 6 shows measurements of transistor mismatch for 30 different transistor sizes operated down to the nanoamperes range.[3] We can see that a 1.2/4 nMOS transistor operating at 1 nA has a current mismatch standard deviation of $\sigma = 7\%$. In our case, we want to achieve a 3-bit precision with the most significant bit. This means that its must be below 1/8 of the range, which implies a standard deviation better than $\sigma \leq 2\%$. Looking at Fig. 6, we can see that there is no transistor size that can achieve this for currents less than or around 1 nA. Consequently, we require some kind of calibration mechanism. In our prototype we used the mini-DACs calibration scheme reported elsewhere [70] at each pixel. This scheme, which results in fairly compact layouts, exploits the use of MOS ladder structures to digitally control the aspect ratio of an equivalent transistor. Fig. 7 shows the schematic of the so-called digi-MOS [70]. When all switches are ON, the ladder structure is equivalent to a MOS transistor of almost $2W/L$ in size. As switches are turned off, the structure is equivalent to a MOS of size wW/L, where w is a digitally controlled weight of value

$$w = \sum_{i=0}^{n} \frac{b_i}{2^{n-i}} \tag{2}$$

where b_i is either 0 or 1. Currents I_{calN} and I_{calP} in Fig. 5 are generated with the circuits shown in Fig. 8. Transistors within the boxes "pixel" are replicated once per pixel, while the rest belong to the periphery and are not replicated. Consequently, currents I_{refn} and I_{refp} are replicated once per pixel and will present mismatch from pixel to pixel. This mismatch is represented on the right side of Fig. 8 when $w_{calN} = w_{calP} = 0$. Each staircase on the right-hand side of Fig. 8 corresponds to one pixel. For example, for I_{calN}, each pixel can store a different value for w_{calN}. This way, after calibration, the I_{calN} values of the pixels should be very close to the thick dotted horizontal line. For optimum calibration, the peripheral current bias values I_{refn} and $I_{\Delta n}$ should be adjusted so that the minimum I_{calN} value for $w_{calN} = 0$ equals the maximum I_{calN} value for maximum w_{calN}. For I_{calP} equivalent considerations are made. In practice, we do not calibrate trying to find very stable values for I_{calP} (or I_{calN}) from pixel to pixel, as shown in Fig. 8. In practice, we do calibrate trying to find stable values for the pixels output frequencies. This way, calibration compensates for mismatch originated by: 1) the replication of currents I_{refp}, $I_{\Delta p}$, I_{refn}, and $I_{\Delta n}$; 2) the mismatch in transistors $M_{n1} - M_{n3}$ and $M_{p1} - M_{p3}$; (c) the mismatch in the total capacitance of pixels

[3]For smaller currents, mismatch tends to stabilize.

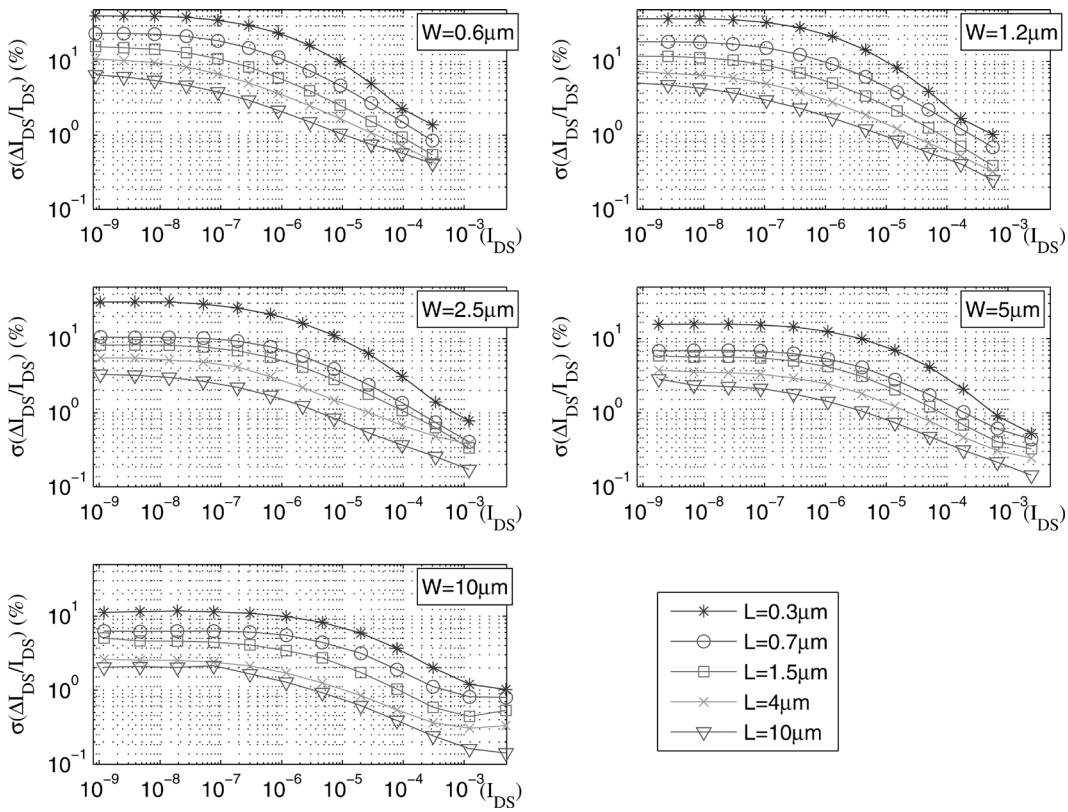


Fig. 6. Measurements of nMOS transistor current mismatch standard deviations for a CMOS 0.35-μm technology. Twenty-five different transistor sizes were characterized, sweeping W from 0.6 to 10 μm and L from 0.3 to 10 μm. Transistor current mismatch standard deviation $\sigma(\Delta I_{\rm DS}/I_{\rm DS})$ is shown on the vertical axes in %. Horizontal axes represent drain current $I_{\rm DS}$. These curves were obtained by sweeping $V_{\rm GS}$ from 0 V to 3.3 V, while maintaining $V_{\rm DS}$ at 1.65 V $(V_{\rm DD}/2)$.

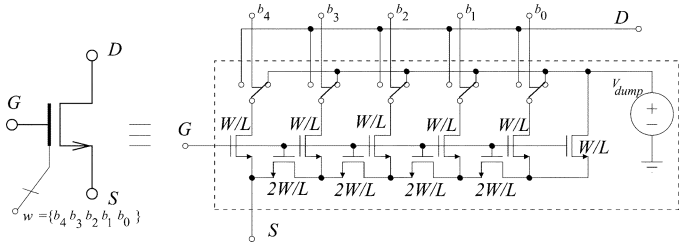


Fig. 7. Schematic diagram of digi-MOS.

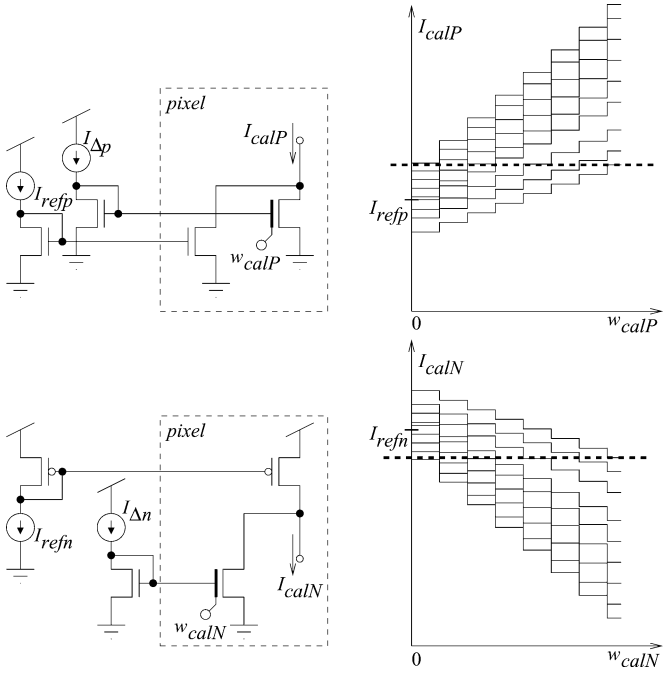


Fig. 8. Digi-MOS-based circuits for generating the in-pixel calibration currents.

node V_c; and (d) the mismatch in offset voltage of all voltage comparators.

Calibration words w_{calN} and w_{calP} are stored on static RAM cells within each pixel, have 5-bit resolution each, do not have a sign bit, and are loaded at startup. Calibration needs to be done for each fabricated chip sample and is done by loading a kernel of size 1×1 of maximum value, sending events of constant frequency for each pixel, and identifying each pixel frequency as a function of the stored calibration word. Calibration words and peripheral bias currents are optimized for minimum dispersion in pixel output frequency values. This procedure is repeated twice to calibrate the two signed parts of current pulsing circuits, by either changing the sign of the input events or the sign of the stored 1×1 kernel.

The block labeled *Logic* in Fig. 5 decides whether a positive or negative current pulse should be provided and what value it should have. Its output directly controls the calibrated pulsing current sources. It also takes care of the forgetting mechanism. This way, the forgetting mechanism implemented in our chip exploits the availability of calibrated current pulses. Therefore, by adding some extra logic, we have an accurate calibrated forgetting mechanism. The details of the *Logic* block are

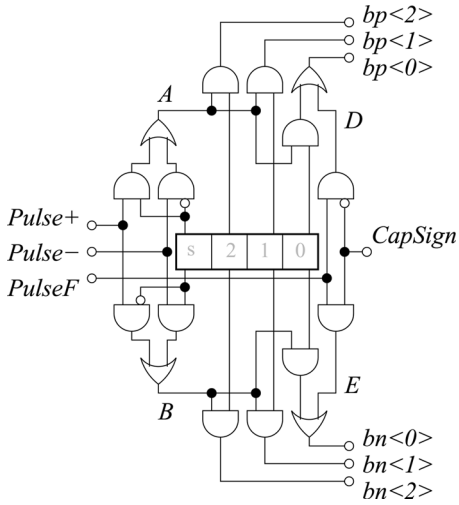


Fig. 9. Schematic diagram of the *Logic* block in Fig. 5.

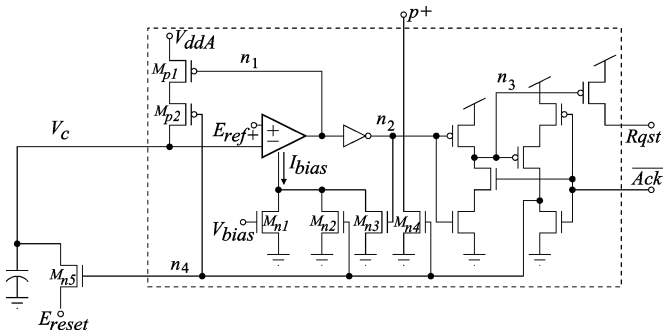


Fig. 10. Detailed schematic diagram of *Positive Event Block* in Fig. 5.

shown in Fig. 9. It contains a dynamic 4-bit weight register that temporarily holds the convolution weight and the sign of the pixel. This is the weight copied from the peripheral static kernel-RAM. During normal event convolution processing signal $PulseF$ is low (no forgetting), and, depending on the sign of the chip input event, the monostable will trigger a pulse through either line $Pulse+$ or $Pulse-$. If the signs of input event and pixel weight are equal, the monostable pulse will appear on node A, and the weight bits $\langle 2 \ldots 0 \rangle$ will come out through lines $\mathrm{bp}\langle 2 \ldots 0 \rangle$, thus producing a positive charge for the integration capacitor in Fig. 5. If the signs of input event and pixel weight are different, the monostable pulse will appear on node B, and the pixel weight will come out through lines $\mathrm{bn}\langle 2 \ldots 0 \rangle$, thus producing a negative charge for the integration capacitor. The forgetting mechanism works by providing global forgetting pulses at input $PulseF$ of constant frequency and width. Both frequency and width are adjusted through configuration registers. The digital controller will take care of providing these forgetting pulses. Two counters are periodically reset. The first one, which controls the frequency of $PulseF$, is decremented by the system clock and the controller checks if it has reached 0, in which case it activates the pulse and the other counter is decremented until '0' to finish the pulse. If the system is idle (no events are being processed), $PulseF$ is triggered when the first counter reaches 0. The forgetting pulses will produce either a positive or negative charge, depending on the sign of the net integration present at the capacitor. If the voltage at the integration capacitor is above V_{reset}, we have a positive net integration ($CapSign = 1$, in Figs. 5 and 9) and negative forgetting charges are produced by activating signal $\mathrm{bn}\langle 0 \rangle$ in Fig. 9. If the integration capacitor voltage is below $V_{\mathrm{reset}}(CapSign = 0)$, then positive forgetting charges are produced by activating signal $\mathrm{bp}\langle 0 \rangle$.

The boxes labeled *Positive Event Block* and *Negative Event Block* in Fig. 5 respectively detect whether the positive threshold $E_{\mathrm{ref}+}$ or the negative one $E_{\mathrm{ref}-}$ are reached by integration capacitor voltage V_c, reset this voltage to E_{reset}, send a row request $Rqst$ to the row arbiter, and, when the row acknowledge Ack comes back, activates either the $p+$ or $p-$ column lines, signaling that a positive or negative event has been generated by this pixel. Fig. 10 shows the detailed circuit schematic diagram for the box *Positive Event Block*. The circuit for the *Negative Event Block* is a symmetrical version. While the integration capacitor is being charged or discharged, the event blocks remain idle. When idle, transistors M_{p1} and M_{n5} in Fig. 10 are off, while M_{p2} is on. The source of M_{p1} is separated from the power supply to minimize leakages while it is off [67], [68]. When V_c approaches $E_{\mathrm{ref}+}$, usually slowly, the comparator output n_1 will slowly turn on M_{p1}. This will enable a positive feedback that will speed up the charging of the capacitor through transistor M_{p1}, making the comparator toggle faster and minimizing the transition time. Also, to speed up this transition, the comparator bias current, which is maintained low through bias V_{bias} during the idle periods for minimum power consumption, is momentarily increased to a high value through transistor M_{n3}. To this end, the inverter between nodes n_1 and n_2 has been carefully designed to present a transition threshold that turns on M_{n3} when n_1 is still at the beginning of its transition. Once n_1 is low and n_2 high, n_3 goes low (note that $\overline{Ack}$ is high during idle periods), sending a row request by pulling up line $Rqst$. After a small delay (a few nanoseconds), the row arbiter will acknowledge back, activating $\overline{Ack}$ to a low value, which will set node n_4 high. This will disconnect the capacitor from V_{DDA} by turning off M_{p2} and will reset its voltage to E_{reset} by turning on M_{n5}. Since we also want the comparator to transition back fast, it will need a high bias current until it finishes its transition back. However, during back transitioning, transistor M_{n3} will be turned off, which will make the comparator slow before the transition actually finishes. For this reason, transistor M_{n2} has been added, so that the comparator is kept fast until the acknowledge signal from the row arbiter returns to its idle state. This way the comparator bias current is kept at a high value until after it has transitioned back to its resting state. Since $\overline{Ack}$ is active for a few nanoseconds, this is not a major power consumption concern. Also, during the $\overline{Ack}$ time is active (and n_4 high), column signal $p+$ is activated, signaling that this column of the arbitrated row has generated a positive event. In a similar way, the *Negative Event Block* would activate signal $p-$, if capacitor voltage V_c reached the negative threshold $E_{\mathrm{ref}-}$.

The right-peripheral row arbitration circuits and top-peripheral column arbitration and output event generation circuits follow the row-parallel event read-out scheme proposed by Boahen in 2000 [24], and we use the same circuits with minor modifications to handle the sign bit as well [65].

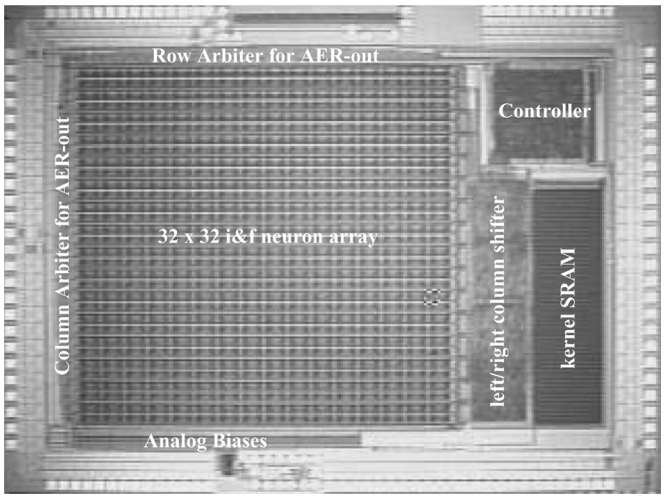


Fig. 11. Microphotograph of fabricated 32 × 32 pixel convolution chip with up to 32 × 32 programmable kernel.

IV. Experimental Results

A small size 4×5 mm^2 prototype chip was fabricated in the AMS 0.35-μm CMOS process. Its microphotograph is shown in Fig. 11. We can see the array of 32 × 32 i&f pixels of approximately 3×3 mm^2 in size, the digital controller of size around 850×900 μm^2, the kernel static RAM of 32 × 32 4-bit words of size 2050×600 μm^2, and the left/right column shifter of size 2150×450 μm^2. We can also see the row and column arbiters of the AER-out section. The remainder of the circuits occupies a small amount of area.

The chip can address an input space of 128 × 128 pixels. A set of configuration registers loaded at startup indicate the corner coordinates of the 32 × 32 pixels within the 128× 128 input address space. The chip can generate output events at a maximum rate of[4] 25×10^6eps (events/s: event communication cycle time of 40 ns).[5] The input events throughput depends on kernel size and internal clock frequency. The event cycle time is given by $(4 + 2 \times n_k)T_{\text{clock}}$, where n_k is the number of programmed kernel lines (from 1 to 32) and T_{clock} is the internal clock period. The internal clock could be set up to 200 MHz ($T_{\text{clock}} = 5$ ns) before observing operation degradation. Maximum input event throughput can therefore vary between 33×10^6 eps (30-ns event cycle time) for a one line kernel down to 3×10^6 eps (340-ns event cycle time) for a full 32 line kernel. The power consumption of the chip depends on input AER throughput and kernel size. For example, for a high input throughput, the power consumption varies between 66–150 mW, depending on kernel size.

The different parameters the user can control are shown in Table I. The top eight are analog parameters, while the two bottom parameters are digital words. Current biases are controlled by digital words in peripheral current DACs. Other digital words loaded at startup are the calibration words for each pixel and the kernel size and weights.

The pixel layout is shown in Fig. 12. The left side shows a complete pixel, with its different components highlighted: the integration capacitor, the calibration circuitry, the pulsing and weighting nMOS and pMOS transistors that feed the capacitor, the logic block of Fig. 9, the set of dynamic weight registers that hold the pixel convolution weight, the comparator used for the forgetting circuitry, and the asynchronous logic for generating the output events (half of it shown in Fig. 10). The area of the active circuitry is 67×78 μm^2, where the routing is not included. The routing lines are shared by the neighbors, by grouping four pixels symmetrically around the center as shown in the right-hand side of Fig. 12. This way, the integrating capacitors of the four pixels, which are the most noise sensitive elements, are placed close together in the center. The most noise sensitive analog lines are routed (like voltage bias lines) close to the capacitors. The dynamic weight registers and the logic block are the noisiest circuits within the pixel and are kept as far as possible from the capacitors. The calibration circuitry is loaded at startup and remains silent throughout normal operation. The AER output asynchronous logic is triggered when the capacitor reaches threshold, and thus does not disturb the capacitor while it is being charged. The pitch of the four pixel layout inside the pixel array is 185.6 μm $\times$ 191.5 μm. Consequently, the effective pixel area, including routing is 92.8×95.8 μm^2.

TABLE I
Biases and Parameters

control parameter	meaning
I_{refn}	global bias for negative charge packets
$I_{\Delta n}$	bias for local calibration of negative charge packets
I_{refp}	global bias for positive charge packets
$I_{\Delta p}$	bias for local calibration of positive charge packets
T_h	charge packet pulse width
E_{reset}	integration capacitor reset level
E^+_{ref}	integration capacitor positive threshold
E^-_{ref}	integration capacitor negative threshold
$(x, y)_{min}$	lower left coordinate of pixel array within the 128x128 total input address space
$(x, y)_{max}$	upper right coordinate of pixel array within the 128x128 total input address space

Next we show some experimental chip measurements, including characterizations and image processing operations. The chip is used and characterized by means of a specially developed AER infrastructure based on FPGAs [71]. From this infrastructure, we used two functional elements: 1) an AER sequencer, which takes artificially generated AER sequences stored in a computer's memory and transforms them into physical AER streams and 2) an AER data logger, which collects AER streams and either visualizes them on a computer screen in real time or stores the data in computer's memory for later analysis.

A. Pixel Characterization

The first measurements consist of characterizing the pixel response to different convolution weights and calibration words. For this, a uniform image was transformed into AER and fed into the chip input AER port. Then, the generated AER output events were collected, the frequency for each pixel identified, and all pixel frequencies represented as a 2-D image. First, the pixel output frequencies were measured without calibration as a

[4]Measured by shorting output $Rqst$ and Ack lines.

[5]An event rate of 25 Meps is a fairly high rate for present state-of-the-art AER chips. For example, in the multi-AER module vision system reported in [64] the event rate was always below 1 Meps at any AER link.

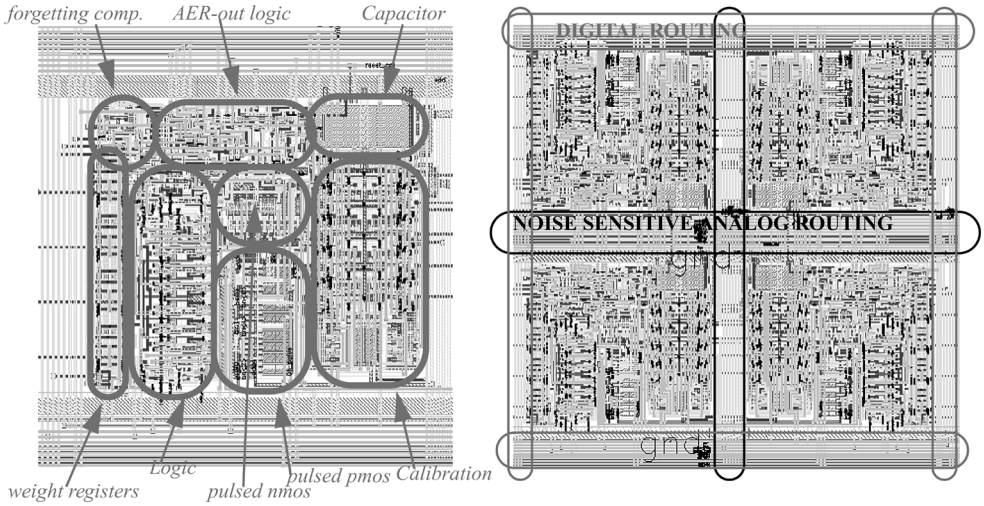


Fig. 12. Layout of i&f neuron pixel, on the left. On the right, four pixels with horizontal and vertical symmetry for sharing common routing lines.

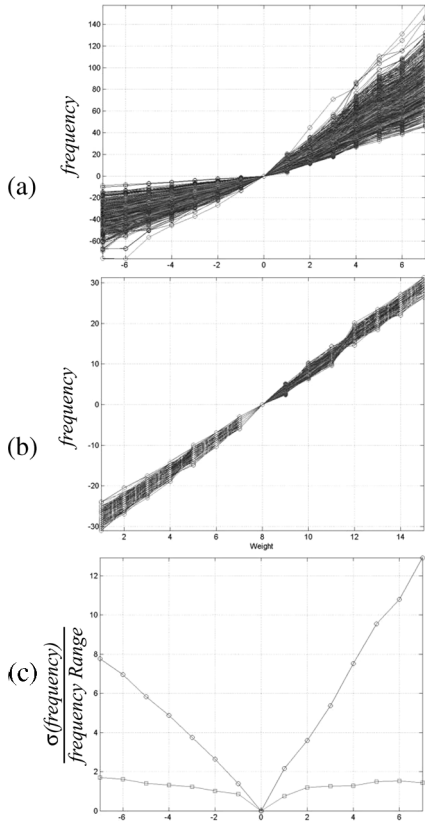
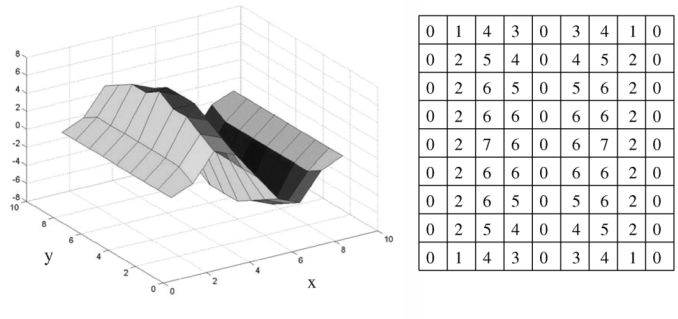


Fig. 13. Measured pixel frequencies as function of convolution weight, for all $32\times32 = 1024$ pixels. (a) Before calibration (constant w_{cal}). (b) After calibration (optimum w_{cal} for each pixel). (c) Computed standard deviations (in %) of measured relative frequency spreads among pixels before and after calibration.

function of convolution weight only. This is shown in Fig. 13(a), where convolution weights were varied from -7 to $+7$, and the measured frequencies of all $32 \times 32 = 1024$ pixels are shown. We can see that, for positive output events, frequencies at maximum convolution weight have a dispersion of 110 Hz over a range of 155 Hz. Consequently, relative spread is 71%. Similarly, for negative output events, frequencies at maximum convolution weight present a spread of over a range of 75 Hz, which corresponds to a relative spread of 87%.



0	1	4	3	0	3	4	1	0
0	2	5	4	0	4	5	2	0
0	2	6	5	0	5	6	2	0
0	2	6	6	0	6	6	2	0
0	2	7	6	0	6	7	2	0
0	2	6	6	0	6	6	2	0
0	2	6	5	0	5	6	2	0
0	2	5	4	0	4	5	2	0
0	1	4	3	0	3	4	1	0

Fig. 14. Gabor kernel for vertical edge extraction.

To calibrate the array of pixels, all pixels were loaded with the same convolution weight and calibration weight, convolution weights were set to either the maximum positive value ($+7$) or to the minimum negative value (-7), and the 5-bit calibration word was swept from 1 to 32. Pixel output frequencies and relative values between $I_{\text{ref}p}$ and $I_{\Delta p}$ (as well $I_{\text{ref}n}$ as and $I_{\Delta n}$) were identified, and for each pixel one positive and one negative calibration word was selected to minimize the frequency spread between all pixels. Once this set of $32\times32\times2 = 2048$ 5-bit calibration words was identified, it was loaded into the pixels calibration static registers, and the pixels frequencies versus convolution weight were measured again. The result is shown in Fig. 13(b). Fig. 13(c) shows the resulting precision as a function of convolution weight. As can be seen, standard deviation after calibration is kept below 2%.

Pixel operation is characterized by a special parameter defined as "*the number of maximum charge packets required to produce an output event*" N_M. According to Fig. 5, the maximum positive current charging up the pixel capacitor is[6] $7I_{calP}$. The voltage increment after one such maximum pulse is $\Delta V_M = 7I_{calP}T_h/C$, where T_h is the pulse width. T_h is usually set to the maximum possible value imposed by the maximum event throughput, which is $(4 + 2 \times n_k)T_{\text{clock}}$, as stated earlier. For maximum clock frequency (200 MHz), T_h can be set between 30 and 340 ns, depending on the number

[6]The maximum negative current is $7I_{calN}$, which, after calibration, should be $7I_{calP}$.

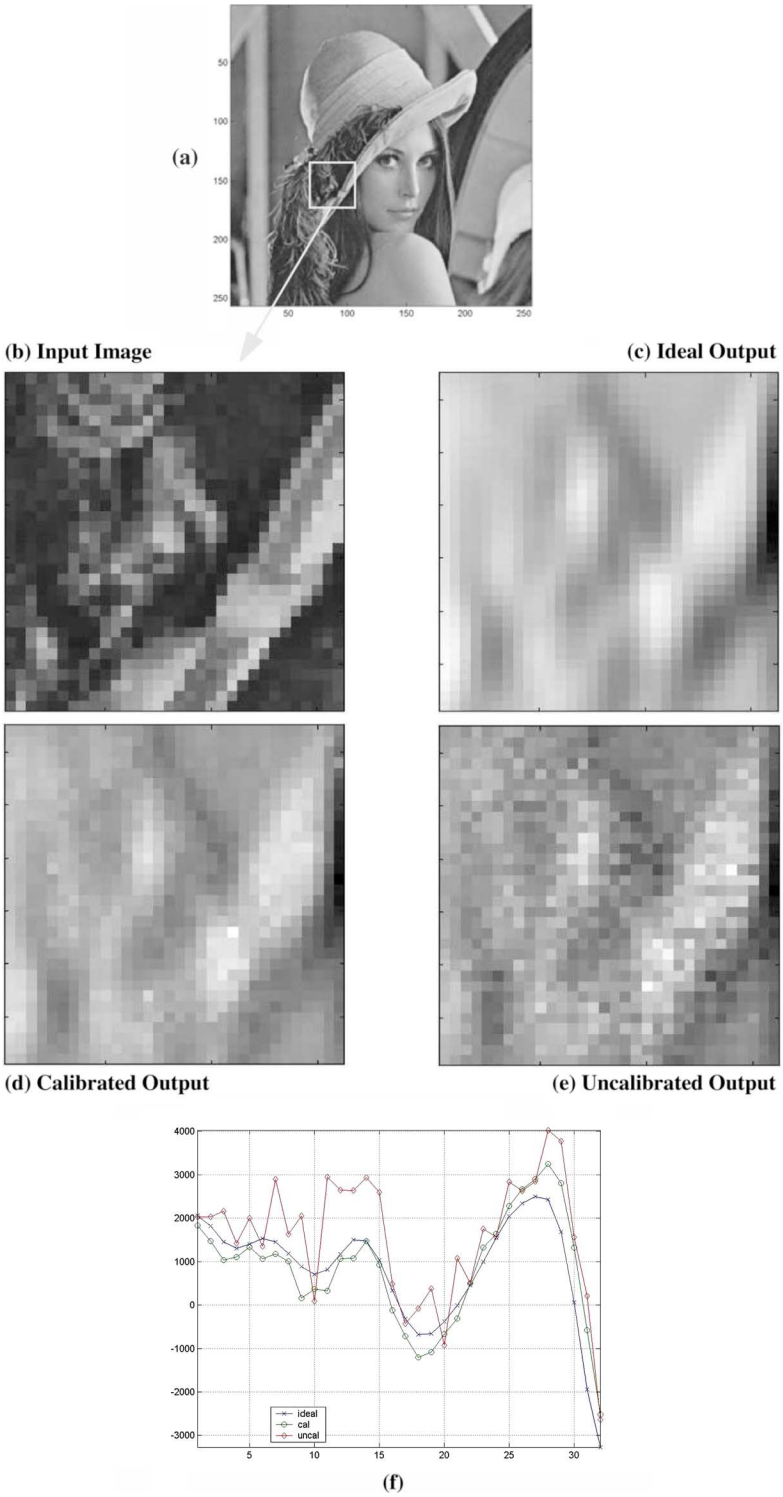


Fig. 15. Experimentally obtained convolution processing results with a Gabor kernel for vertical edge extraction. (a) Real image from which we use a 32 × 32 subimage. (b) Selected input subimage from (a). (c) Ideal convolution output computed with MATLAB. (d) Experimentally obtained convolution output with calibration ON. (e) Experimentally obtained convolution output with calibration OFF. (f) Numerical representation of pixel frequency for one of the rows (row 21 starting from bottom) obtained from (c), (d), and (e).

of kernel lines $n_k \in [1, 32]$. If the pixel capacitor only receives maximum current pulses of the same sign, then we can compute N_M as

$$N_M = \left| \frac{E^+_{\text{ref}} - E_{\text{reset}}}{\Delta V_M} \right| = C \frac{\left| E^+_{\text{ref}} - E_{\text{reset}} \right|}{7 |I_{calP}| T_h}. \quad (3)$$

Pixel capacitor has a value of about $C = 140$ fF, $|E^+_{\text{ref}} - E_{\text{reset}}| = |E_{\text{reset}} - E^-_{\text{ref}}|$ is usually set around 0.5 V, and I_{calP} can be set over a very wide range (from 10 pA to 100 μA). This allows the user to adjust a very wide range for N_M: from one to several hundred thousand maximum pulses. In a practical situation, where multiple convolutions are cascaded in a multilayer

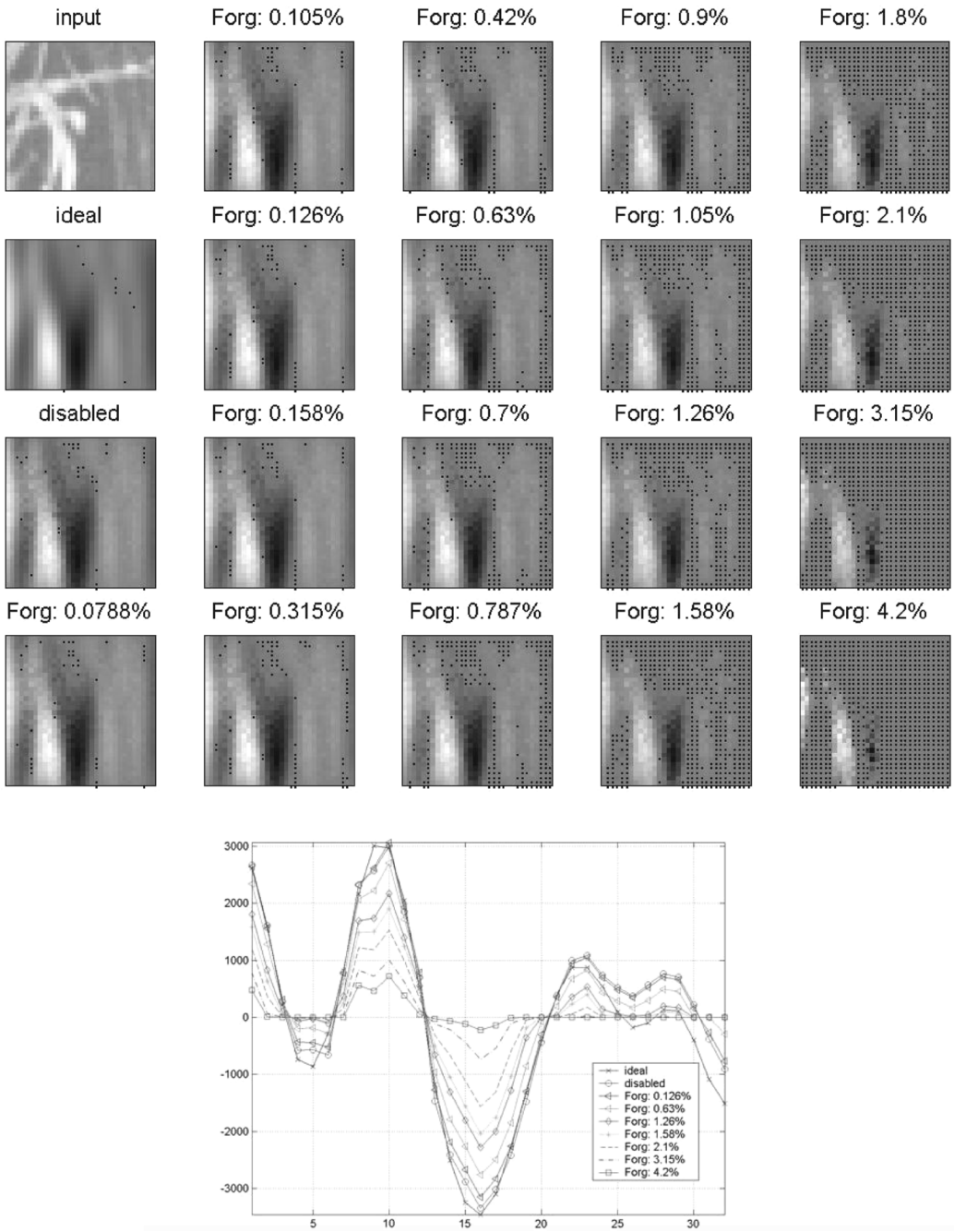


Fig. 16. Convolution with forgetting. The mosaic on the top shows the input image, the ideal output image, and 18 different outputs, each with a different forgetting ratio. The first output was obtained with forgetting disabled, while the rest were set to the forgetting ratio indicated on top of each image. Forgetting ratios are expressed in percent $(100 \times t_{pf}/T_f)$. In the lower part, the experimentally obtained pixel frequency for some of the forgetting ratios is shown.

AER cortical system, it is convenient to keep the same maximum pixel firing frequency for all stages. This means that if input event maximum pixel frequency is $f_{\max}$, then convolution output pixel maximum frequency should also be $f_{\max}$. Consequently, N_M should be adjusted according to kernel shape: if an input image is presented that produces the maximum possible convolution output, the most active output pixels should fire events also at $f_{\max}$. For example, if kernel size is 32×32 with all weights at maximum value (+7), and input image is a 32×32 square of maximum value (input pixels fire at $f_{\max}$), then the central pixel receives, during a time period of $1/f_{\max}$, $32 \times 32 = 1024$ maximum pulses, and should generate output events at a rate of $f_{\max}$. Therefore, N_M should be 1024. For the example 9×9 kernel in Fig. 14 if a perfect edge with half pixels at $f_{\max}$ and the other half at $-f_{\max}$ is centered with the kernel, then $N_M = (1/7)\sum |w_{ij}| \approx 32$, where $\sum |w_{ij}|$ is the sum of all of the kernel weights (in absolute value) shown in Fig. 14.

During the calibration procedure described above, we set $N_M = 7$ (by setting $I_{calN} \approx I_{calP} \approx 2.5$ nA, $T_h = 300$ ns, $|E_{\text{ref}}^{+} - E_{\text{reset}}| = |E_{\text{reset}} - E_{\text{ref}}^{-}| = 0.6$ V), and a kernel of size 1×1 with $w_{11} = 7$. Frequency of input events was 200 Hz.

B. Convolution Processing of Images

This chip can be programmed to compute convolutions with kernels of arbitrary size and shape, as long as size is less than or equal to 32 × 32 pixels and shape is discretized to a four bit resolution (including sign). To illustrate convolution processing, we programmed the 9 × 9 Gabor kernel shown in Fig. 14, used typically in image processing for orientation extraction. As input image we selected a real photograph [shown in Fig. 15(a)] and selected from it the 32 × 32 subimage shown in Fig. 15(b). Fig. 15(c) shows the result of the mathematical computation (with MATLAB) of this input image with the kernel in Fig. 14. The images shown in Fig. 15(d) and (e) are obtained experimentally from the convolution chip by collecting the AER output stream and reconstructing it in a computer. The image in Fig. 15(d) corresponds to the output obtained when calibration is turned ON and Fig. 15(e) when it is OFF. The numerical frequency values for row 21 (row count starts from bottom) are shown in Fig. 15(f), comparing the cases of calibrated and uncalibrated outputs with respect to the ideal output.

C. Convolution Processing With Forgetting

The effect of introducing a loss (or forgetting) mechanism in the convolution integration is equivalent to adding a threshold for the outputs. This is quite obvious, because now the integral of the weighted incoming events within a given time interval has to overcome the effect of the loss. The experimental confirmation of this can be seen in Fig. 16. The forgetting rate is adjusted, through the chip configuration registers, by setting the period T_f of the global forgetting signal ($PulseF$ in Figs. 5 and 9) and its width t_{pf}. For the results in Fig. 16, t_{pf} was set constant and equal to 630 ns, while T_f was changed between infinity (no forgetting) and 15 μs. The forgetting ratio is defined as t_{pf}/T_f. In the top part of Fig. 16, a mosaic with different images is shown. The top left image corresponds to the input image used for this experiment, which is also a 32 × 32 subframe of the photograph in Fig. 15(a). Convolutions are computed using the kernel in Fig. 14. The image below this one is the mathematically computed ideal output, and the rest are chip outputs for different forgetting ratios from 0 (disabled) to 0.042. The gray levels code from maximum positive output (white) to minimum negative output (black). Consequently, zero output activity is coded by an intermediate gray level. The pixels with zero activity have been marked with a small black dot. As forgetting ratio is increased, we can see there are larger areas with zero output activity, while only those areas with activity above a positive threshold or below a negative threshold yield nonzero outputs. The bottom part in Fig. 16 shows the numerical pixel frequency values measured experimentally along row 21 for some of the forgetting ratios. Input events were coming in with a pixel maximum frequency of 2.1 kHz. We can see that increasing the forgetting ratio not only introduces a thresholding effect, but also a global attenuation of the output signals. This is because the average current integrated onto the capacitor $\overline{I}_c$ is the semi-rectified version of the difference between the average pulsing current $\overline{\Sigma I}$ produced by the input events and the average forgetting current I_f. This can be seen in Fig. 17(a), where as the forgetting current is increased a central gap appears, and the net average capacitor current $\overline{I}_c$ decreases. To compensate for this, the charging currents produced by the input events can be scaled by a factor α. This is illustrated in Fig. 17(b), where we can see that the maximum current $\overline{I}_c$ (and therefore, maximum output frequency) is kept constant. The data shown in Fig. 16 were obtained following the scheme in Fig. 17(a), without compensation $(\alpha = 1)$.

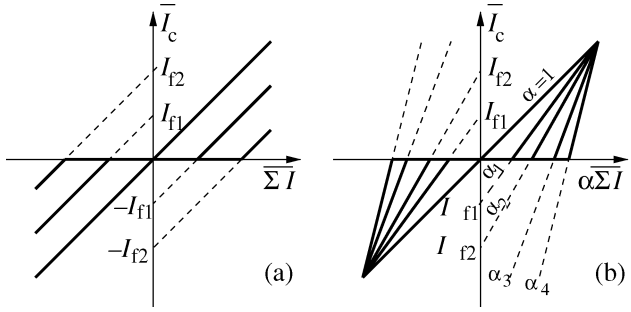


Fig. 17. Visualization of thresholding effect of forgetting mechanism. (a) I_{cal} is kept constant while forgetting current is increased, thus producing a reduction of net capacitor current and attenuation of output frequency. (b) To compensate for this effect, currents I_{cal} are scaled by a factor α to maintain constant output frequency.

D. Convolution With Forgetting for Spatio-Temporal Correlated Pattern Detection

An interesting feature of processing convolutions with lossy integration or forgetting, is that it allows for spatio-temporal correlated pattern detection. This means that in order to recognize a given object shape, the features forming this pattern have to appear within a given time interval. To illustrate this we will program the convolution chip to detect a circumference of radius 12 pixels. Fig. 18(a) shows the 32 × 32 kernel programmed for this operation. Note that the kernel is such that for pixels on the circumference of radius 12 the kernel has value 2, while for the rest the kernel has value −1. The kernel has 120 pixels of value +2. This kernel, when convolved with the input image in Fig. 18(b), provides an output image with all pixels saturated at the minimum negative value except for the center of the circumference which will be maximum positive. In this experiment, the 120 pixels of the circumference were firing at a frequency of 2.5 kHz, and the active output pixel was firing at the same frequency. The value of N_M for the pixels was set to 34.3. If we now present the input image as two half circles[7] [see Fig. 18(c), (d)], the output of the convolution will be different depending on whether or not forgetting is turned ON. If forgetting is turned ON, the output of the chip is the one shown in Fig. 18(e), which means that the chip does not recognize the two half circles sequenced in time as a single circumference. However, if forgetting is turned OFF, the chip output is shown in Fig. 18(f) and the chip does recognize the two halves as a single circumference. Consequently, forgetting allows the system to discriminate whether the features of a figure appear within a certain time frame, so that they can be considered as being originated by the presence of the figure and not by the sequential presence of two or more separate pieces of a figure. This way,

[7]During a time interval of 100 ms, only pixels of one half circumference were activated. During the next 100-ms interval, the pixels of the other half were activated, and so on.

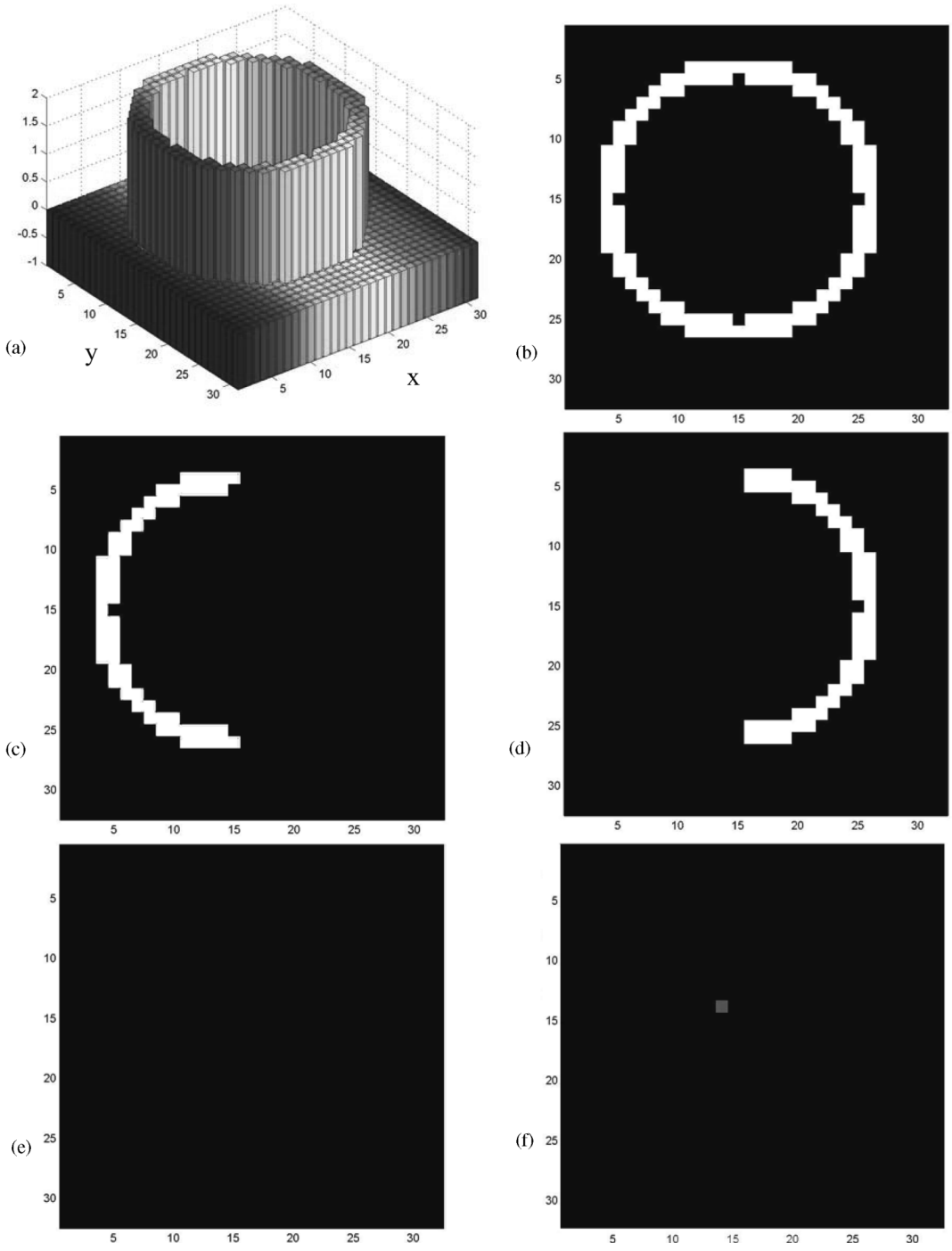


Fig. 18. Spatio-temporal pattern recognition by performing convolution with lossy integration (forgetting). (a) Circumference kernel of radius ten pixels. (b) Circumference input image of radius ten pixels. (c) Left half of input circle and (d) right half of input circle. (e) Output with forgetting ON. (f) Output with forgetting OFF.

the convolution chip processing is equivalent to a coincidence detector.

E. Recognition of Rotating Propellers

An experiment that demonstrates the high-speed processing capabilities of AER based systems is the recognition of high-speed rotating propellers. For this, we will feed the convolution chip with a stimulus consisting of two rotating propellers. Each propeller has a different shape, as shown in Fig. 19. One is rectilinear, and the other has an S-like shape. When the propellers rotate at high speed, one only sees a solid circle that moves slowly across the screen. Therefore, a human observer would not be able to discriminate between the two propellers. In this experiment, we generated an artificial sequence of events representing the rotating propellers. This sequence of events was generated numerically as explained in the Appendix and physically provided in real-time by a sequencer PCB [72], [73] connected

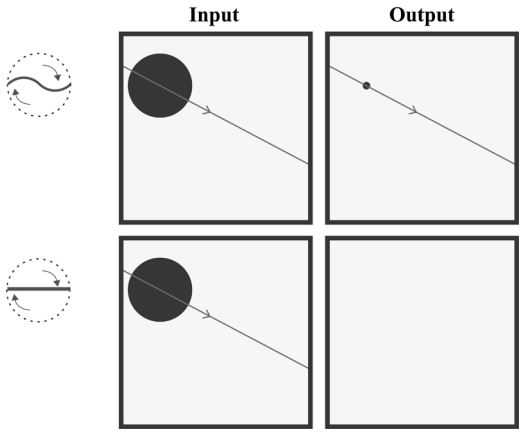


Fig. 19. Response of convolution chip to two rotating propellers of different shape. The top row corresponds to an S-shaped propeller, while the bottom row corresponds to a rectilinear propeller. The kernel programmed onto the chip is for recognizing the S-shaped propeller when it is in horizontal position. The left-hand columns show the input stimuli. Since the propellers are rotating at high speeds, only solid circles are seen. The right-hand columns show the output of the convolution processing. The top output detects the S-shaped propeller in horizontal position. Consequently, in the top output a dot would be seen moving along the screen, which means that the s-shaped propeller is being followed. The bottom output is empty, since the convolution chip does not detect an S-shaped propeller.

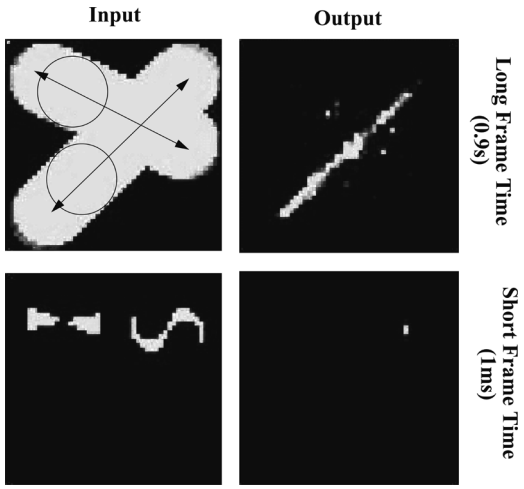


Fig. 20. Real-time recognition and monitoring of high speed (100 revolutions/s) simultaneous rotating propellers. Left-hand frames are inputs, and right-hand frames are outputs of convolution processing. Top frames are the result of collecting events during 0.9 s, while bottom frames are the result of collecting events for 1 ms (1/10 of a revolution).

to the input AER port of our convolution chip.[8] The input and output AER ports of the convolution chip were recorded simultaneously using a monitor PCB [72], [73]. All input and output events, conveniently time-stamped, were recorded in computer memory for careful analysis. Note that AER streams are not represented by sequences of frames (as in conventional video). However, to show the results graphically, we will collect events during certain time intervals, and show 2-D images. This is, for

[8]Note that we had to generate this stimulus artificially because presently there is no AER motion retina capable of correctly sensing propellers rotating up to 5000 revolutions per second. For example, the AER motion retina in [36] is able to sense rotations of up to 400–500 revolutions per second.

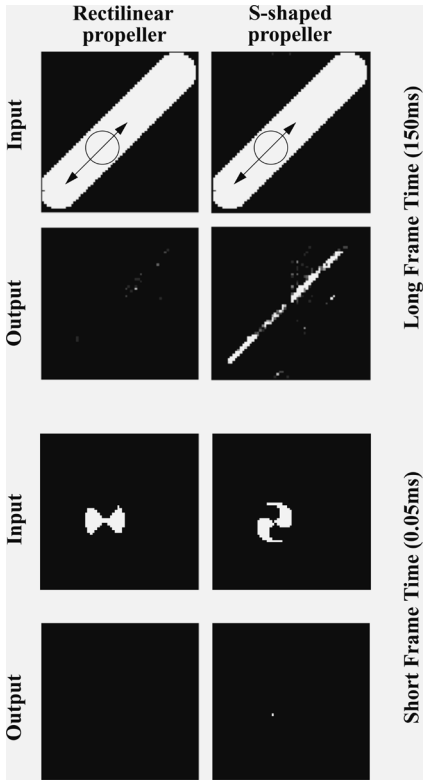


Fig. 21. Real-time recognition and monitoring of high speed (5000 revs/sec) rotating propellers. Left-hand frames: rotating propeller is rectilinear. Right-hand frames: rotating propeller is S-shaped. Top four frames were collected during 150 ms. Bottom four frames were collected during 50 μs (1/4 of a propeller revolution).

example, what we show in Figs. 20 and 21. Fig. 20 corresponds to an experiment where two propellers rotating at 100 revolutions per second (6-k revolutions per minute) move across the screen and intersect at a given point. One propeller is rectilinear and the other is S-shaped. Each propeller has a diameter of 16 pixels. The frames shown have 48 pixels in width and height.[9] There are four frames shown in Fig. 20. Those on the left-hand side correspond to the input stimulus (the rotating propellers), and those on the right-hand side to the output. The bottom ones were generated collecting events during a short time interval of 1 ms (which corresponds to 1/10 of one propeller revolution), while the top ones correspond to a much longer time interval of 0.9 s. The convolution chip was programmed with a kernel to detect the center of the S-shaped propeller when it is in the horizontal position. As can be seen, the output of the convolution chip follows the center of the S-shaped propeller as it moves across the screen. In this experiment forgetting ratio was set to 0.13, $I_{calN} \approx I_{calP} \approx 2.2$ nA, and $N_M = 11$. The kernel loaded for this experiment is shown in Fig. 22, where weights of value -3, $+3$, and $+7$ (from dark to light) were used.

Fig. 21 shows the results of a similar experiment, but now the propellers are rotating at 5000 revolutions per second (300 k revolutions per minute). The programmed kernel is also given by the drawing in Fig. 22, but with weights of value -1, 0, and

[9]Although each convolution chip has an array of 32 $\times$ 32 pixels, remember that the input address space it can process is 128 $\times$ 128.

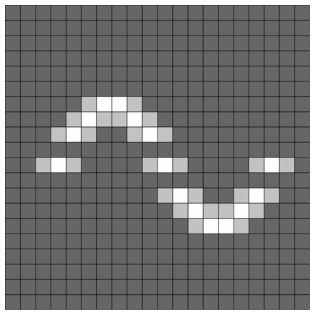


Fig. 22. Kernel used for the rotating propeller recognition experiments. For the propeller experiment rotating at 100 revolutions per second, the kernel weight values were -3, $+3$, and $+7$, while for the propeller experiment rotating at 5000 revolutions per second the kernel weight values were -1, 0, and $+6$.

$+6$ (from dark to light). The input stimulus is either the rectilinear propeller or the S-shaped propeller. The four bottom frames were generated using a 50-μs time interval (1/4 of a propeller revolution), while the four top ones used a time interval of 150 ms (one complete back-and-forth screen crossing). The left-hand frames correspond to the case of the input rectilinear propeller and the right-hand frames to the case of the input S-shaped propeller. As can be seen, the convolution chip correctly follows the S-shaped propeller, while for the rectilinear propeller only some spurious noise is observed at the output. In this experiment, the forgetting ratio was set to 0.0025, $I_{calN} \approx I_{calP} \approx 0.8$ nA, and $N_M = 30$, and an array 2×2 of convolution chips was assembled with a total of 64×64 i&f pixels.

In a practical setup, for example, a motion AER image sensor would be required. For example, the one presented in [36] is able to sense rotating images at up to 400–500 Hz. If the size of the propellers is not known *a priori* (for example, if the distance to the imager can change), then several convolution chips are required in parallel, each programmed with kernels of different size [64].

Note that using conventional frame-based image processing methods to discriminate between the two propellers is a complicated task, which requires a high computational load. First, images must be acquired with an exposure time of at least 100 μs. The recorded propeller may appear with any angle, so the convolution kernels would have to be processed for different orientations (for example, ten different orientations). All this must be performed in real time.

V. Conclusion

We have presented an AER-based transceiver chip that computes convolutions on its 2-D input AER data stream. The pixels can also be programmed to allow a forgetting rate, which allows for thresholding and coincidence detections. The kernel of the convolutions can be programed with 4-bit resolution and can be up to 32×32 pixels in size. The chip can receive AER input data representing images of up to 128×128 pixels, although its output space is only 32×32 pixels. The chip can be tiled in a matrix fashion, so that an array of $N \times M$ chips can provide an output space of $32N \times 32M$ pixels. The chip can handle an input event throughput of between 3 and 33 Mega-events per second, depending on programmed kernel size, and is capable of generating output events at a maximum rate of 25 Mega-events per second. Pixel operation is based on integrating programmable current pulses, using special techniques which allow to program current pulses down to picoamperes within nanoseconds delays. Extensive experimental measurements have been provided, including real-time image processing and recognition tasks, including 2-D image filtering operations, coincidence detections, and recognition/discrimination of propellers rotating at speeds of up to 5000 revolutions per second. The presented AER programmable-convolution/programmable-forgetting rate transceiver chip is a key element for the future development of multilayer hierarchically structured artificial cortical systems for performing complex cognitive tasks. This work has been developed in the context of the EU funded CAVIAR (IST-2001-34124) project, where a multi-AER-module vision demonstrator was assembled [64], which includes the convolution chip reported in this paper, an AER motion sensing retina [36], a 2-D AER Winner-Takes-All module [74], a hebbian-based associative learning module [75], and a set of chip–chip and chip–computer AER interfaces [72], [73].

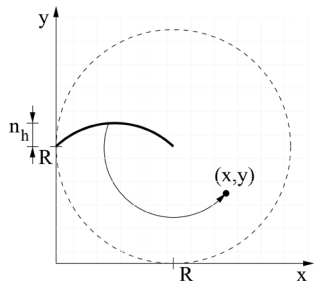


Fig. 23. Diagram of half propeller rotating counterclockwise.

Appendix

To artificially generate the stream of input events representing a propeller (rectilinear or S-shaped) rotating at $1/T_{\rm rot}$ revolutions per second, we used the following procedure. Consider the half propeller rotating counter clockwise shown in Fig. 23 having size R and height n_h. This rotating half propeller sweeps the circular area of radius R. Consider now the grid of pixels inside the square $(x, y) \in [0, 2R]^2$. Every time the rotating half propeller intersects the center of a pixel (x, y), an event for that pixel coordinate is produced at the time of this intersection. Consequently, this event is represented by $(t_{\rm event}, x, y)$. We wrote a little MATLAB script that returns the time it takes for the half propeller in Fig. 23 to intersect the center of a pixel (x, y), given R, $T_{\rm rot}$, and n_h : `tevent = event_time(Trot,nh,R,x,y)`. If a pixel is outside the circle of radius R, it would return `-1`, indicating that no event with this coordinate was produced. This way, one full rotation of the half propeller would produce as many events as there are pixels inside the circle of radius R. For the propellers in Figs. 20 and 21, we used $R = 8$. Consequently, there are about $\pi R^2 \approx 200$ events produced for each half-propeller revolution. The following MATLAB script shows how to generate the events for a half propeller full revolution.

```
function events = geteventsPP(Trot, nh, R)

for x = 1 : 2 * R
    for y = 1 : 2 * R
        tt0(y + 2 * R * (x - 1)) =
        event_time(Trot, nh, x, y, R);
        xt0(y + 2 * R * (x - 1)) = x;
        yt0(y + 2 * R * (x - 1)) = y;
    end
end
indx0 = find(tt0 > -0.5);
tt1 = tt0(indx0);
xt1 = xt0(indx0);
yt1 = yt0(indx0);
[tmp ii] = sort(tt1);
events = [tt1(ii); xt1(ii); yt1(ii)];
```

This script sweeps the (x, y) coordinates of the pixels containing the half propeller full revolution. For each pixel, the function `event_time` is called, which provides the timestamp for that pixel event. Afterwards, the pixels not producing any events are discarded, and the rest are ordered in time. For generating a full propeller, the events of the half propeller are separated into the first half time $(0 < t < T_{\rm rot}/2)$ and second half time $(T_{\rm rot}/2 < t < T_{\rm rot})$. For the events of the first half time we add $T_{\rm rot}/2$ to each timestamp, and for the events of the second half time we subtract $T_{\rm rot}/2$ from each timestamp. Then we merge all events of the two half-time propellers and reorder them according to their timestamps. This is performed by the following MATLAB script.

```
function events_full_propeller =one_rot_propeller
(Trot, nh, R)

events1 = getevents(Trot, nh, R);
indx0 = find(events1(1,:) > Trot/2);
events2A = events1(:, indx0);
events2A(1,:) = events2A(1,:) - Trot/2;
LT = length(events1);
LA = length(indx0);
events2B = events1(:, 1 : LT - LA);
events2B(1,:) = events2B(1,:) + Trot/2;
events2 = [events2A, events2B];
ee = [events1, events2];
[tmp ii] = sort(ee(1,:));
events_full_propeller =[ee(1, ii); ee(2, ii); ee(3, ii)];
```

If we now want to add motion to the rotating propeller, the procedure is simply to add a trajectory vector $\overrightarrow{r}_c = (x_c, y_c)$ describing the motion of the central point of the propeller. In general, if $\overrightarrow{r}_o$ is the initial propeller center position and $\overrightarrow{v}(t) = (v_x(t), v_y(t))$ is its instantaneous velocity vector, then the propeller center trajectory vector is given by

$$\overrightarrow{r}_c(t) = \overrightarrow{r}_a + \int_0^t \overrightarrow{v}(t)dt. \tag{4}$$

In the case of a rectilinear motion of constant speed, v_x and v_y are constant, and the trajectory vector (x_c, y_c) would be given by

$$\begin{aligned} x_c(t) &= x_o + v_x t \\ y_c(t) &= y_o + v_y t. \end{aligned} \tag{5}$$

The values calculated for (x_c, y_c) are then added to the x- and y-coordinates of the rotating propeller events. This is done by the following MATLAB script.

```
function ee2 =
moving_propeller(Trot, nh, R, Nrevs, xo, yo, vx, vy)

ees = one_rot_propeller(Trot, nh, R);
ee2 = ees;
for i = 1 : Nrevs - 1
    ees(1,:) = ees(1,:) + Trot;
    ee2 = [ee2, ees];
end
ee2(2,:) = ee2(2,:) + (xo - R) + round(vx * ee2(1,:));
ee2(3,:) = ee2(3,:) + (yo - R) + round(vy * ee2(1,:));
```

The new input arguments are the number of propeller revolutions (`Nrev`), the starting coordinate for the propeller center (`xo`, `yo`), and the rectilinear speed (`vx`, `vy`). First, the script generates the events for the number of propeller revolutions specified. Afterwards, it applies (5) to these events. The new x- and y-coordinates are rounded to the nearest integer. The value for time t in (5) is taken from the timestamp of each event.

REFERENCES

[1] H. Fujii, H. Ito, K. Aihara, N. Ichinose, and M. Tsukada, "Dynamical cell assembly hypothesis—theoretical possibility of spatio-temporal coding in the cortex," *Neural Netw.*, vol. 9, pp. 1303–1350, 1996.

[2] G. A. Orban, *Neural Operations in the Visual Cortex*. Berlin, Germany: Springer-Verlag, 1984.

[3] M. Shadlen and W. T. Newsome, "Noise, neural codes and cortical organization," *Current Opinion Neurobiol.*, vol. 4, pp. 569–579, 1994.

[4] G. M. Shepherd, *The Synaptic Organization of the Brain*, 3rd ed. Oxford, U.K.: Oxford Univ. Press, 1990.

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

[6] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recogn.*, vol. 15, pp. 455–469, 1982.

[7] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Netw.*, vol. 1, pp. 119–130, 1988.

[8] ——, "Analysis of the process of visual pattern recognition by the neocognitron," *Neural Netw.*, vol. 2, pp. 413–420, 1989.

[9] Y. Le Cun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 255–258.

[10] C. Neubauer, "Evaluation of convolution neural networks for visual recognition," *IEEE Trans. Neural Netw.*, vol. 9, no. 4, pp. 685–696, Jul. 1998.

[11] M. Matsugu, K. Mori, M. Ishi, and Y. Mitarai, "Convolutional spiking neural network model for robust face detection," in *Proc. 9th Int. Conf. Neural Inf. Process. (ICONIP'02)*, 2002, vol. 2, pp. 660–664.

[12] B. Fasel, "Robust face analysis using convolutional neural networks," in *Proc. Int. Conf. Pattern Recogn. (ICPR'02)*, 2002, pp. 40–43.

[13] M. Browne and S. S. Ghidary, "Convolutional neural networks for image processing: an application in robot vision," in *Advances Artif. Intell.: Proc. 16th Australian Conf. on AI*, Nov. 2003, pp. 641–652.

[14] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jiménez, and B. Linares-Barranco, "An arbitrary kernel convolution AER-transceiver chip for real-time image filtering," in *Proc. IEEE Int. Conf. Circuits Syst. (ISCAS'06)*, 2006, pp. 3145–3148.

[15] M. Sivilotti, "Wiring considerations in analog VLSI systems with application to field-programmable networks," Ph.D. dissertation, Comp. Sci. Div., California Inst. Technol., Pasadena, CA, 1991.

[16] M. Mahowald, "VLSI analogs of neural visual processing: A synthesis of form and function," Ph.D. dissertation, Comp. Sci. Div., California Inst. Technol., Pasadena, CA, 1992.

[17] ——, *An Analog VLSI Stereoscopic Vision System*. Boston, MA: Kluwer, 1994.

[18] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Silvilotti, and D. Gillespie, "Silicon auditory processors as computer peripherals," *IEEE Trans. Neural Netw,*, vol. 4, no. 3, pp. 523–528, May 1993.

[19] J. P. Lazzaro and J. Wawrzynek, "A multi-sender asynchronous extension to the address-event protocol," in *Proc. 16th Conf. Advanced Res. VLSI*, W. J. Dally, J. W. Poulton, and A. T. Ishii, Eds., 1995, pp. 158–169.

[20] A. Mortara and E. A. Vittoz, "A communication architecture tailored for analog VLSI artificial neural networks: Intrinsic performance and limitations," *IEEE Trans. Neural Netw.*, vol. 5, no. 3, pp. 459–466, Jul. 1994.

[21] A. Mortara, E. A. Vittoz, and P. Venier, "A communication scheme for analog VLSI perceptive systems," *IEEE J. Solid-State Circuits*, vol. 30, no. 6, pp. 660–669, Jun. 1995.

[22] Z. Kalayjian and A. G. Andreou, "Asynchronous communication of 2-D motion information using winner-takes-all arbitration," *Int. J. Analog Integr. Circuits Signal Proc.*, vol. 13, no. 1–2, pp. 103–109, Mar./Apr. 1997.

[23] K. A. Boahen, "Communicating neuronal ensembles between neuromorphic chips," in *Neuromorphic Systems Engineering: Neural Networks in Silicon*, T. S. Lande, Ed. Norwell, MA: Kluwer, 1998, ch. 11.

[24] ——, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 5, pp. 416–434, May 2000.

[25] ——, "A burst-mode word-serial address-event link—Part I: Transmitter design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 7, pp. 1269–1280, Jul. 2004.

[26] ——, "A burst-mode word-serial address-event link—Part II: Receiver design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 7, pp. 1281–1291, Jul. 2004.

[27] ——, "A burst-mode word-serial address-event link—Part III: Analysis and test results," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 7, pp. 1292–1300, Jul. 2004.

[28] 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.

[29] 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, no. 12, pp. 2325–2333, Dec. 2003.

[30] 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, no. 2, pp. 160–172, Feb. 2002.

[31] C. Shoushun and A. Bermak, "A low power CMOS imager based on time-to-first-spike encoding and fair AER," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'05)*, 2005, pp. 5306–5309.

[32] X. G. Qi, X. , and J. Harris, "A time-to-first-spike CMOS imager," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, Vancouver, BC, Canada, 2004, pp. 824–827.

[33] M. Azadmehr, J. Abrahamsen, and P. Häfliger, "A foveated AER imager chip," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'05)*, Kobe, Japan, 2005, pp. 2751–2754.

[34] J. Kramer, "An on/off transient imager with event-driven, asynchronous read-out," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'02)*, Phoenix, AZ, 2002, pp. 165–168.

[35] P. Lichtsteiner, T. Delbrück, and J. Kramer, "Improved on/off temporally differentiating address-event imager," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, Vancouver, BC, Canada, 2004, pp. 211–214.

[36] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 dB 30 mW asynchronous vision sensor that responds to relative intensity change," in *IEEE ISSCC Dig. Tech. Papers*, San Francisco, CA, 2006, pp. 508–509.

[37] M. Arias-Estrada, D. Poussart, and M. Tremblay, "Motion vision sensor architecture with asynchronous self-signaling pixels," in *Proc. 7th Int. Workshop Comput. Architecture Machine Perception (CAMP'97)*, 1997, pp. 75–83.

[38] 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.

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

[40] G. Indiveri, A. M. Whatley, and J. Kramer, "A reconfigurable neuromorphic VLSI multi-chip system applied to visual motion computation," in *Proc. Int. Conf. Microelectron. Neural, Fuzzy Bio-Inspired Syst. (Microneuro'99)*, Granada, Spain, 1999, pp. 37–44.

[41] K. Boahen, "Retinomorphic chips that seez quadruple images," in *Proc. Int. Conf. Microelectron. Neural, Fuzzy Bio-Inspired Syst. (Microneuro'99)*, Granada, Spain, 1999, pp. 12–20.

[42] R. Z. Shi and T. K. Horiuchi, "A VLSI model of the bat dorsal nucleus of the lateral lemniscus for azimuthal echolocation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'05)*, Kobe, Japan, 2005, pp. 4217–4220.

[43] A. van Schaik and S.-C. Liu, "AER EAR: a matched silicon cochlea pair with address event representation interface," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'05)*, Kobe, Japan, 2005, pp. 4213–4216.

[44] G. Cauwenberghs, N. Kumar, W. Himmelbauer, and A. G. Andreou, "An analog VLSI chip with asynchronous interface for auditory feature extraction," *IEEE Trans. Circ. Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 5, pp. 600–606, May 1998.

[45] M. Oster and S.-C. Liu, "Spiking inputs to a spiking winner-take-all circuit," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, vol. 18, pp. 1051–1058 [Online]. Available: http://books.nips.cc/papers/files/nips18/NIPS2005_0521.pdf, (NIPS'06)

[46] J. Abrahamsen, P. Häfliger, and T. S. Lande, "A time domain winner-take-all network of integrate-and-fire neurons," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, Vancouver, BC, Canada, May 2004, vol. V, pp. 361–364.

[47] E. Chicca, G. Indiveri, and R. J. Douglas, "An event-based VLSI network of integrate-and-fire neurons," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, Vancouver, BC, Canada, 2004, vol. V, pp. 357–360.

[48] T. Teixeira, A. G. Andreou, and E. Culurciello, "Event-based imaging with active illumination in sensor networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'05)*, Kobe, Japan, 2005, pp. 644–647.

[49] P. Häfliger, "Asynchronous event redirecting in bio-inspired communication," in *Proc. ICECS01*, 2001, pp. 87–90.

[50] D. H. Goldberg, G. Cauwenberghs, and A. G. Andreou, "Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons," *Neural Netw.*, vol. 14, no. 6–7, pp. 781–793, 2001.

[51] R. J. Vogelstein, U. Mallik, and G. Cauwenberghs, "Silicon spike-based synaptic array and address-event transceiver," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, 2004, vol. 5, pp. 385–388.

[52] R. J. Vogelstein, F. Tenore, R. Philipp, M. S. Adlerstein, D. H. Goldberg, and G. Cauwenberghs, "Spike timing-dependent plasticity in the address domain," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, vol. 15.

[53] R. J. Vogelstein, U. Mallik, G. Cauwenberghs, E. Culurciello, and R. Etienne-Cummings, "Saliency-driven image acuity modulation on a reconfigurable silicon array of spiking neurons," in *Advances in Neural Information Processing Systems*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, vol. 17, pp. 1457–1464.

[54] U. Mallik, R. J. Vogelstein, E. Culurciello, R. Etienne-Cummings, and G. Cauwenberghs, "A real-time spike-domain sensory information processing system," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'05)*, Kobe, Japan, 2005, pp. 1919–1922.

[55] R. J. Vogelstein, U. Mallik, E. Culurciello, R. Etienne-Cummings, and G. Cauwenberghs, "Spatial acuity modulation of an address-event imager," in *Proc. 11th IEEE Int. Conf. Electron., Circuits Syst. (ICECS '04)*, Dec. 2004, pp. 207–210.

[56] S.-C. Liu, J. Kramer, G. Indiveri, T. Delbrück, and R. Douglas, "Orientation-selective aVLSI spiking neurons," *Neural Netw.*, vol. 14, pp. 629–643, 2001.

[57] P. Vernier, 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.

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

[59] T. Y. W. Choi, B. E. Shi, and K. Boahen, "An ON-OFF orientation selective address event representation image transceiver chip," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 2, pp. 342–353, Feb. 2004.

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

[61] S. Grossberg, E. Mingolla, and J. Williamson, "Synthetic aperture radar processing by a multiple scale neural system for boundary and surface representation," *Neural Netw.*, vol. 8, no. 7/8, pp. 1005–1028, 1995.

[62] S. Grossberg, E. Mingolla, and W. D. Ross, "Visual brain and visual perception: How does the cortex do perceptual grouping?," *Trends Neurosci.*, vol. 20, pp. 106–111, 1997.

[63] E. Mingolla, W. Ross, and S. Grossberg, "A neural network for enhancing boundaries and surfaces in synthetic aperture radar images," *Neural Netw.*, vol. 12, no. 3, pp. 499–511, 1999.

[64] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, H. Kolle Riis, T. Delbrück, S. C. Liu, S. Zahnd, A. M. Whatley, R. Douglas, P. Häfliger, G. Jimenez-Moreno, A. Civit, T. Serrano-Gotarredona, A. Acosta-Jiménez, and B. Linares-Barranco, "AER building blocks for multi-layers multi-chips neuromorphic vision systems," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, vol. 18, pp. 1217–1224 [Online]. Available: http://books.nips.cc/papers/files/nips18/NIPS2005_0268.pdf, (NIPS'06)

[65] R. Serrano-Gotarredona, T. Serrano-Gotarredona, and B. Linares-Barranco, "Event generators for address event representation transmitters," in *Proc. SPIE*, Jun. 2005, vol. 5839, Bioengineered and Bioinspired Systems, pp. 148–159.

[66] B. Linares-Barranco, T. Serrano-Gotarredona, R. Serrano-Gotarredona, and J. Costas-Santos, "A new charge-packet driven mismatch-calibrated integrate-and-fire neuron for processing positive and negative signals in AER based systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, May 2004, vol. 5, pp. 744–747.

[67] B. Linares-Barranco and T. Serrano-Gotarredona, "On the design and characterization of femtoampere current-mode circuits," *IEEE J. Solid-State Circuits*, vol. 38, no. 8, pp. 1353–1363, Aug. 2003.

[68] B. Linares-Barranco, T. Serrano-Gotarredona, R. Serrano-Gotarredona, and C. Serrano-Gotarredona, "Current-mode techniques for sub-pico ampere circuit design," *Int. J. Analog Integr. Circuits Signal Process.*, vol. 38, pp. 103–119, 2004.

[69] T. Serrano-Gotarredona and B. Linares-Barranco, "CMOS mismatch model valid from weak to strong inversion," in *Proc. Eur. Solid State Circuits Conf. (ESSCIRC'03)*, Sep. 2003, pp. 627–630.

[70] B. Linares-Barranco, T. Serrano-Gotarredona, and R. Serrano-Gotarredona, "Compact low-power calibration mini-DACs for neural massive arrays with programmable weights," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1207–1216, Sep. 2003.

[71] A. Linares-Barranco, G. Jimenez-Moreno, B. Linares-Barranco, and A. Civit-Ballcels, "On algorithmic rate-coded AER generation," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 771–788, May 2006.

[72] F. Gomez-Rodriguez and R. Paz-Vicente *et al.*, "AER tools for communications and debugging," in *Proc. IEEE Int. Symp. Circuits Syst., (ISCAS'06)*, May 2006, pp. 3253–3256.

[73] R. Paz-Vicente and A. Linares-Barranco *et al.*, "PCI-AER interface for neuro-inspired spiking systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'06)*, May 2006, pp. 3161–3164.

[74] S.-C. Liu and M. Öster, "Feature competition in a spike based winner-take-all VLSI network," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'06)*, May 2006, pp. 3634–363637.

[75] P. Häfliger, "Adaptive WTA with an analog VLSI neuromorphic learning chip," *IEEE Trans. Neural Netw.*, submitted for publication.

[76] R. Serrano-Gotarredona *et al.*, "On real-time AER 2D convolutions hardware for neuromorphic spike based cortical processing," *IEEE Trans. Neural Netw.*, submitted for publication.

[77] K. A. Zaghloul and K. Boahen, "Optic nerve signals in a neuromorphic chip: Part I and II," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 4, pp. 657–675, Apr. 2004.

[78] J. Costas Santos *et al.*, "A contrast retina with on-chip calibration for neuromorphic spike-based AER vision systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, accepted for publication.

Rafael Serrano-Gotarredona received the B.S. degree in telecommunications engineering from the University of Seville, Sevilla, Spain, in 2002. He is currently working toward the Ph.D. degree at the "Instituto de Microelectrónica de Sevilla" (IMSE-CNM-CSIC).

During June and July of 2006, he was with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, as a Visiting Scholar. He has also visited the Neuroinformatics Institute of ETH, Switzerland, in the context of the European-funded research project CAVIAR. His research interests include analog and mixed-signal VLSI circuit design applied to vision processing systems and high-speed LVDS chip communications.

Mr. Serrano-Gotarredona is the recipient of a scholarship from the Spanish Ministry of Education and Science.

Teresa Serrano-Gotarredona received the B.S. degree in electronic physics and the Ph.D degree in VLSI neural categorizers from the University of Seville, Sevilla, Spain, in 1992 and 1996, respectively, and the M.S. degree from the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD, in 1997.

She completed her doctoral research at the "Sevilla Microelectronics Institute" (IMSE), which is one of the institutes of the "National Microelectronics Center" (CNM) of the "Spanish Research Council" (CSIC) of Spain. She was on a sabbatical stay at the Electrical Engineering Department, Texas A&M University, College Station, during the spring of 2002. She was an Assistant Professor with the University of Seville from 1998 until 2000. Since June 2000, she has held a Tenured Scientist position with the "Sevilla Microelectronics Institute" (IMSE), Sevilla, Spain. Since January 2006, she is also a part-time Professor at the university of Seville. 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, and real-time vision processing chips. She is coauthor of the book *Adaptive Resonance Theory Microchips* (Kluwer, 1998).

Dr. Serrano-Gotarredona was corecipient of the 1997 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS Best Paper Award for the paper "A real-time clustering microchip neural engine" and of the IEEE CAS Darlington Award for the paper "A General Translinear Principle for Subthreshold MOS Transistors." While with the Johns Hopkins University, she was supported by a Fulbright Fellowship.

Antonio Acosta-Jiménez received the "Licenciado en Física" degree and the Ph.D. degree in electronic physics from the University of Seville, Sevilla, Spain, in 1989 and 1995, respectively.

He is currently with the Institute of Microelectronics of Seville, CNM-CSIC, and with the Department of Electronics and Electromagnetism, University of Seville, where he has been an Associate Professor since 1998. His current research interests are in the areas of CMOS digital and mixed-signal VLSI design, low-power and low-noise CMOS, description of timing phenomena in VLSI digital systems, and asynchronous and self-timed circuits. He has authored or coauthored more than 70 international scientific publications and has been involved in different National and European R&D projects. He was the General Ghair of the 2002 PATMOS International Workshop.

Bernabé Linares-Barranco received the B.S. degree in electronic physics, the M.S. degree in microelectronics, and the Ph.D. degree in high-frequency OTA-C oscillator design from the University of Seville, Sevilla, Spain, in 1986, 1987, and 1990, respectively, and the Ph.D. degree in analog neural network design from Texas A&M University, College Station, in 1991.

Since September 1991, he has been a Tenured Scientist with the Sevilla Microelectronics Institute (IMSE), which is one of the institutes of the National Microelectronics Center (CNM) of the Spanish Research Council (CSIC) of Spain. In January 2003, he was promoted to Tenured Researcher and in January 2004, to Full Professor of Research. Since March 2004, he is also a part-time Professor at the University of Seville. From September 1996 to August 1997, he was on sabbatical with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD, as a Postdoctoral Fellow. During spring 2002, he was a Visiting Associate Professor with the Electrical Engineering Department, Texas A&M University. He has been involved with circuit design for telecommunication circuits, very large scale integration (VLSI) emulators of biological neurons, VLSI neural-based pattern recognition systems, hearing aids, precision circuit design for instrumentation equipment, bio-inspired VLSI vision processing systems, transistor parameter mismatch characterization, address-event-representation VLSI, RF circuit design, and real-time vision processing chips.

Dr. Linares-Barranco was corecipient of the 1997 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS Best Paper Award for the paper "A real-time clustering microchip neural engine" and the 2000 IEEE Circuits and Systems Darlington Award for the paper "A general translinear principle for subthreshold MOS Transistors." He organized the 1994 Nips Post-Conference Workshop "Neural Hardware Engineering". From July 1997 until July 1999, he was an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART II, ANALOG DIGITAL AND SIGNAL PROCESSING, and, since January 1998, he has also been an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS. He was Chief Guest Editor of the 2003 IEEE TRANSACTIONS ON NEURAL NETWORKS Special Issue on Neural Hardware Implementations. He is coauthor of the book *Adaptive Resonance Theory Microchips* (Kluwer, 1998). He was the coordinator of the EU-funded CAVIAR (Convolution AER Vision Architecture for Real-Time) Project.