A USB3.0 FPGA Event-based Filtering and Tracking Framework for Dynamic Vision Sensors

A. Linares-Barranco [1,3], F. Gómez- Rodríguez [3], V. Villanueva [2], L. Longinotti [2], T. Delbrück [1,2].
[1] Institute of Neuro Informatics. UZH-ETHZ and [2]IniLabs GmbH, Zurich, SWITZERLAND
[3] Robotic and Technology of Computers Lab. University of Seville. Seville, SPAIN
alinares@atc.us.es

***Abstract*—Dynamic vision sensors (DVS) are frame-free sensors with an asynchronous variable-rate output that is ideal for hard real-time dynamic vision applications under power and latency constraints. Post-processing of the digital sensor output can reduce sensor noise, extract low level features, and track objects using simple algorithms that have previously been implemented in software. In this paper we present an FPGA-based framework for event-based processing that allows uncorrelated-event noise removal and real-time tracking of multiple objects, with dynamic capabilities to adapt itself to fast or slow and large or small objects. This framework uses a new hardware platform based on a Lattice FPGA which filters the sensor output and which then transmits the results through a super-speed Cypress FX3 USB microcontroller interface to a host computer. The packets of events and timestamps are transmitted to the host computer at rates of 10 Mega events per second. Experimental results are presented that demonstrate a low latency of 10us for tracking and computing the center of mass of a detected object.**

I. Introduction

Dynamic Vision Sensors [1] mimic part of the biological retina's functionality in silicon chips using CMOS circuits with asynchronous communication of sensor output events (called Address-Event-Representation) [2][3]. Each sensing unit, or pixel in these chips, is analogous to an ON/OFF channel pair of bipolar and ganglion cells after the cones of a biological retina. They work independently of each other. Each of them senses incoming visual information and sends events that signal log intensity brightness changes. The main advantages of these sensors is the combination of local gain control, low latency, and sparse output. Due to the circuitry, a sensed change by any of the pixels is sent out within a millisecond of occurrence and events have microsecond resolution at the chip output. This philosophy is radically different to the one in classic machine vision. Conventional digital cameras work by integrating photocurrent over a short period of time (exposure time) and then they spend a considerable amount of clock cycles in sending out a whole picture or frame, even though only a few pixels may have changed since the last captured frame. In the central nervous system, most important information arrives first to the brain and it is processed in a high priority way, sometimes in an involuntary and reflexive way. The use of digital cameras to emulate such kind of processing implies an enormous amount of computational resources to extract that crucial information from the frames in the available time between consecutive frames. For hard real-time systems this biological mimicking approach is crucial because it is allowing the development of embedded systems working with visual information that are able to perform relatively complex visual tasks, like fast object detection and tracking, as we propose in this paper. Several research groups have developed similar sensors, but the operational principle is the same, like ATIS/cnmDV [1]. The processing framework proposed here could apply to other kinds of event-based sensors, like cochlea [4] and olfactory ones [5]. Novel event-based processing systems exploit mesh networks of convolutions [6] that allow the implementation of ConvNets for event-based classification; and architectures able to implement event-based learning, like MINITAUR [7]. This paper is focused on cleaning and improving the stream of events coming out from the DVS with the aim of reducing unnecessary communication and increasing the accuracy and speed of any event-based classifier or learning algorithm using these sensors. We present a new framework for event-based algorithms in hardware that uses the DAVIS[11] retina camera and a low cost FPGA in the same platform. A complete logic infrastructure is presented covering not only the inclusion of any event-based filter or algorithm, but also the interface to a FX3 microcontroller that supports the USB3.0 bus to a personal computer able to send packets of events at a peak rate of 10Meps with precise timestamps that can be monitored and stored through the jAER open-source software [8]. Section II presents a completely event-based processor able to filter background noise and then detect and track fast and slow objects. Then section III describes the hardware framework to implement and test event-based visual algorithms. Finally, some results and conclusions are shown in sections IV and V.

II. FPGA Event-based Filtering and Object Tracking

Figure 1 shows the proposed filter architecture for multiple objects detection and tracking. It is composed of two main blocks: the background activity filter (BAF) and the object tracker which offers the center of mass calculation (CMCell). The CMCell can be replicated several times in daisy chain.

This research is supported in part by Spanish grant (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02), by the Swiss NCCR Robotics and EU projects SEEBETTER and VISUALISE.

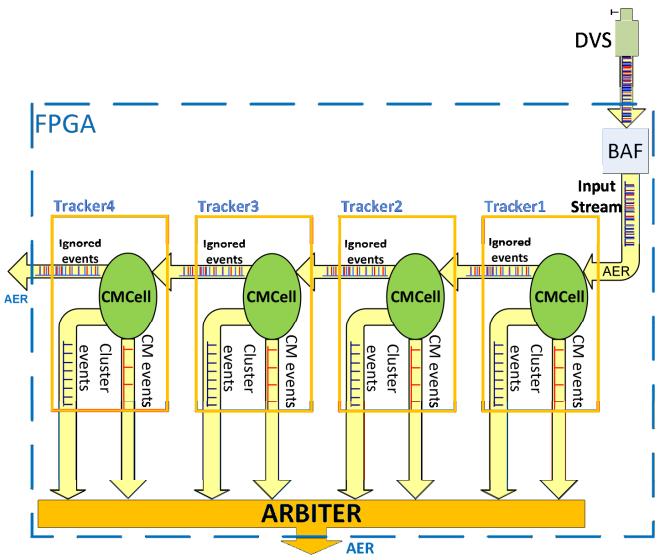

Figure 1. BAF and x4 Object Trackers (in cascade) block diagram.

A. *Background Activity Filter*

The ON/OFF bipolar events of the DAVIS, which represent the temporal contrast changes (DVS events), are processed first by a Background Activity Filter, BAF (jAER[8]). This is able to filter all the non-spatially and non-temporally correlated event rate activity. These DVS streams usually contain sporadic events due to noisy currents not related to light changes in their pixels. This sporadic activity decreases the performance of any filter, so it must be removed. BAF uses an always-on 32-bit timer for time measuring. For each incoming event the current timer value is copied into a *t1* register. A 128x128 array of 32-bit words (implemented on FPGA RAM blocks) stores the last correlated event timestamps (called *t0*, which holds the last *t1* for a set of neighbor's previous events). This 128x128 array can completely hold a sensor with the same size, or a higher resolution sensor by subsampling the DVS address space (in this work 4 neighbor pixels of DAVIS share the same *t0*) as in [9], which implies spatial correlation. BAF will pass through any incoming event if $(t1\text{-}t0) < t_{TH}$, where t_{TH} can be parameterized. Once the decision is made to pass through the event or cancel it, the *t1* value is stored in memory as *t0* to process the next event. It can be stored not only in the corresponding address (nb0), but also in a neighbor-hood (nb1-nb8) to increase the spatial correlation (see Figure 2).

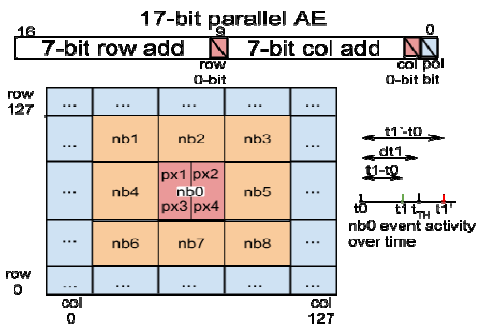

Figure 2. Background Activity Filter. Top: from AE space 7 most significant row and column addresses bits are used to address BAF memory. Center side: 4 pixels (px1-px4) share t0. Max 6x6 pixels spatial correlation can be used. Right side: timestamp t0 corresponds to last spatially correlated event and t1,t1' are two possible current event timestamps around the threshold t_{TH}.

B. *Event-based Tracker*

A tracker must be able to detect a potential object from DVS events and then it has to be able to follow that object while it is moving through the visual field. To accomplish this, each proposed tracker starts waiting for an object at a particular initial position and window of the visual field, called a *cluster*. This initial cluster location and size is configurable by software. As soon as a number of events, N_{ev}, fall into the cluster within a configurable period of time, then, the tracker marks itself to have detected an object. A configurable spatial extension over the cluster size is always monitored by the tracker for dynamic decision making on cluster movements and updates of the cluster size. N_{ev} can be adjusted dynamically for automatic adaptation to different object speeds as explained below. Depending on the evolution of events falling in the cluster several tasks are performed in parallel: (1) The cluster can move through the visual field according to the calculation of the center of mass (CM) of those N_{ev} events. (2) The cluster size can be enlarged or shrunk from its initial size depending on the presence of events both in the cluster and its extension. If there is activity in this extension area the cluster is enlarged. If the activity is concentrated around the center of the cluster then it is shrunk. (3) The current CM can be averaged with a power of 2 fraction of the last CM events to low-pass-filter and reduce big changes in the trajectory. (4) This N_{ev} is fixed when tracker is reset, but if the next N_{ev} events are detected in a very short period of time (the object speed is increasing), the tracker will increase N_{ev} for the next iteration of the process to increase the precision of the CM calculation; conversely if the needed time for receiving N_{ev} events increases (the object speed is decreasing), the tracker will decrease N_{ev} to reduce the latency in the CM calculation (precision can be reduced). These dynamic changes of N_{ev} allow the tracker to adapt to object speed changes automatically. In this work we use an x2 factor for increasing N_{ev} and a /2 factor for decreasing N_{ev}, which are translated into bit-shift operations in the logic.

Up to 4 different objects can be detected and tracked in parallel with the developed architecture. Each of these trackers has one input and three different outputs: CM events (center of mass events), cluster events (detected object events) and pass-through events. This last one is sending out all the events not falling into the cluster of the current tracker, so this output represents the portion of the sensed DVS events where all the events of the first detected object have been filtered out. This output is used by the next tracker for detecting a new object in the visual field. There is no limit in the number of trackers to be implemented except for the resources limit of the selected FPGA. For a Lattice LFE3-17EA (around $40) 10% of resources (1.7k gates) are needed per tracker.

III. Framework

The BAF and trackers have been implemented in a new platform developed as a host platform for any event-based processing algorithm for DVS sensors suitable for FPGA that improve previous state of the art, like [10]. The architecture, as can be seen in Figure 3, is composed of a DAVIS [11] sensor, a Lattice ECP3 FPGA with 17K logic gates, an ADC, an Inertial Motion Unit (IMU) and a Cypress FX3 USB 3.0 Super-Speed microcontroller. The IMU is used for measuring movements of the sensor [12]. It is composed of a gyroscope

and an accelerometer. An on board ADC is converting the analog scan output of the sensor in order to reconstruct a digital frame in the host computer. All the information is converted into events and a timestamp is assigned as needed for the data in the FPGA logic. A CAVIAR [13] connector allows to connect to this framework an external event source, like a sequencer [14].

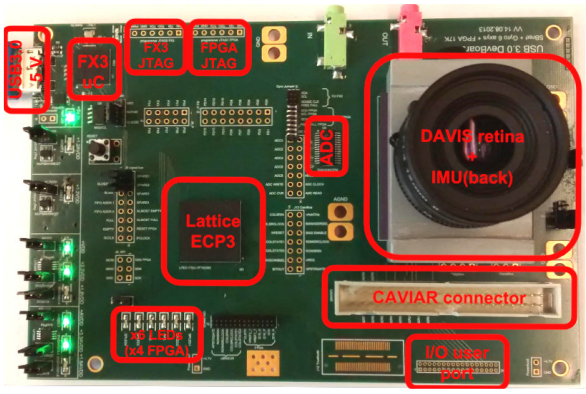

Figure 3. DevBoard USB3 platform photograph. Red marks signalize connectors, LEDs, microcontroller, FPGA, ADC, IMU and DAVIS retina

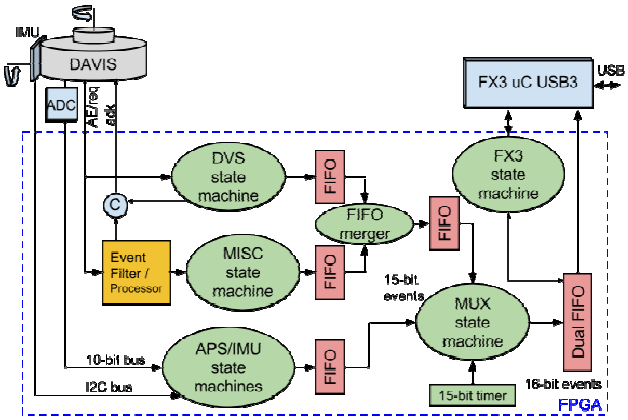

Figure 4. FPGA logic block diagram

Figure 4 shows a block diagram of the logic in the FPGA. The DAVIS is producing three different sensing outputs: (1) Polarized events to identify increments or decrements in the brightness of a particular pixel, called DVS events; (2) periodic sequences of analog values that corresponds to the gray level luminosity of each pixel (APS); and (3) inertial information about the accelerations that the sensor is experiencing (IMU). The DVS output of the DAVIS is connected in parallel to the DVS state machine (SM) and to the Event Filter, with a C-element taking care of the acknowledge (ACK). DVS SM takes the 8-bit events that encode the row or column address together with a bit to indicate whether it corresponds to a row or a column and a polarity bit. Then, it expands the address to 12 bits to support a unified address format even with expected sensor size changes in the future, and a 3-bit header to clearly identify these 12 bits as a DVS event. On the other hand, MISC SM adds a different header to the output of the event-based filter, so these MISC events can be differentiated in the host computer (ie in jAER [8]). These two SMs send their events to two small FIFOs (16 words) that are joined into a next one. In parallel, the APS/IMU SMs are scanning digital frames and inertial information and converting it into a sequence of 15-bit events, as DVS and MISC events but with different header codes. Then, the MUX SM multiplexes all events waiting in FIFOs and, following a priority, a timestamp is attached to some of them using a global and common 15-bit timer. Then they are sent to a double-clocked FIFO (dual-FIFO), to cross into the USB clock domain. The FX3 SM sends those events and time-stamps from the dual-FIFO to the USB buffer of the micro-controller to be sent to the host computer.

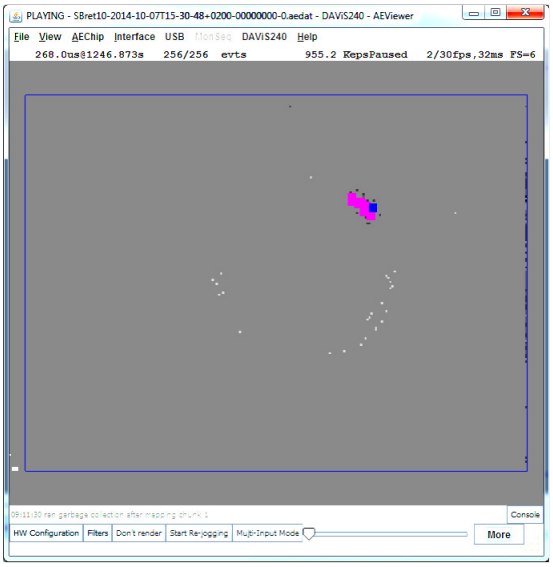

Figure 5. jAER capture of DVS + Tracker cluster events (purple) + Center of Mass events for 268us of a spinning dot at 100rpm.

Figure 5 shows a jAER screen shot of 268us of DVS and MISC events recording while DAVIS was stimulated by a spinning dot at 100rpm using a fan. Black and white dots represent DVS events, while pink dots are those DVS events inside one tracker cluster and blue dots are CM events.

IV. Experiments

Using this framework, with the BAF and x4 trackers in the FPGA logic *"Event/Filter Processor"* block of Figure 4, we have tested the system with two different stimuli, keeping the same initial parameters (Table 1).

A. *Slow speed objects*

Previously recorded events from a 128x128 DVS retina, which was set on a bridge over a 5-lane freeway monitoring many cars, have been used to test this framework for low speed objects. These recorded events can be downloaded from jAER web page [8]. Figure 6 shows the output for one tracker (one car) and 1.2 seconds. Events falling inside the clusters represent a moving car (blue), and CM events are in red. When the car is far away, only a few events are produced at the horizon line (around row address 50). The closer the car is to the bridge, the bigger it becomes, and so the more events are produced by the DVS sensor (lower row addresses). Figure 6A represents the output of a tracker without dynamic adaptation of N_{ev}. When the car is closer (so bigger and faster motion),

CM output is not precise. Figure 6B shows the output with dynamic capabilities. N_{ev} is increased at the same time the car is approaching, and the CM output is more precise. CM inter event interval time is 150 ms when the car is far and 10 ms when the car is closer. N_{ev} is fixed to 15 events for the static version and it was initialized to 5 in the dynamic one, where it oscillates between 10 when the car is far away and 80 events when it is closer.

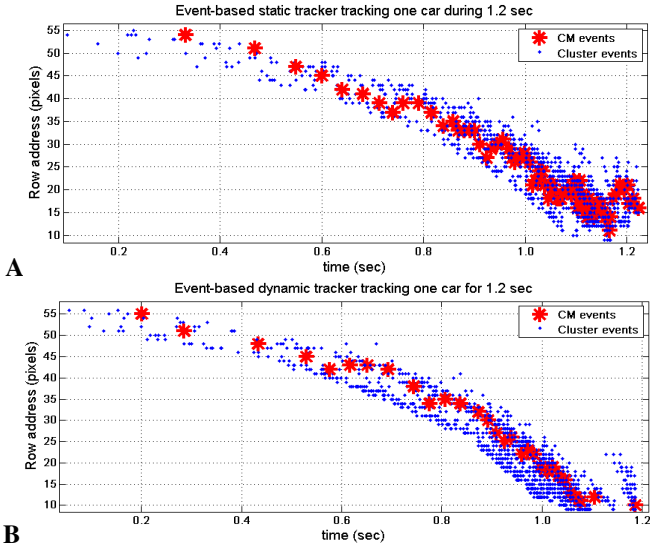

Figure 6. Car tracking using BAF and x4 trackers. Only 1 tracker output is shown. **A:**Without dynamic capabilities. **B:** with dynamic adaptation. Blue dots represent events that fall in the cluster. Red dots represent CM events.

B. *High speed objects*

In this second experiment a set of 52 poker cards was riffled in front of the DAVIS sensor in 500ms, as a riffle shuffler mixes the cards. Regular frame-based digital cameras cannot capture proper images to be processed in this scenario unless they use a frame rate of over 200Hz. With DVS events no special care or operation is needed. The same cluster radius and extension size trackers parameters (Table 1) are used in both experiments. Figure 7 shows the output of the tracker for one card falling from top to bottom of the retina visual field. At the beginning the card is still and when the finger releases the card, it speeds up quickly. It can be seen how the number of events in the cluster per time unit keep growing over time (blue dots), while number of CM events per time unit starts to adjust to the increment of speed and stays in the same range. At the beginning CM inter-event interval is 750 us, and then it is adjusted at 25 us. N_{ev} has a reset value of 3 events and it grows up to 12 to accommodate the stimulus.

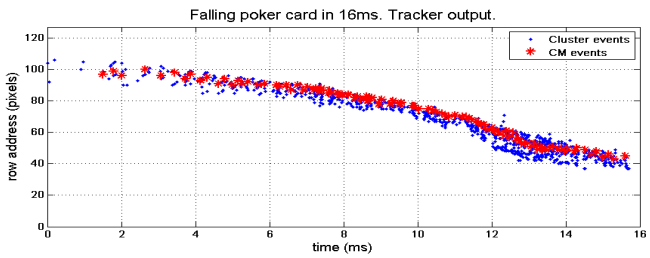

Figure 7. Poker card fast tracking for 16 ms with a Dynamic tracker.

Table 1. Dynamic trackers initial parameters

Parameter	Value
BAF t_{TH}	300 us
Cluster radius	5 pixels
Cluster extension size	2 pixels
Initial Integration (N_{ev})	5 evs (cars), 3 evs (cards)
Inactivity Reset time	200 ms (cars), 2 ms (cards)
CM history average	4 CM events

V. Conclusion

This paper presents an FPGA based framework for event-based filtering and processing, able to detect and track objects at different speeds due to dynamic adaptation of key parameters. It cannot ensure the number of objects in the scene, since a fixed number of trackers are working in parallel on the FPGA, and those represent the maximum detected number of objects. The presented framework has been designed and manufactured as a testing and development platform for event-based algorithms for FPGAs. It has been interfaced to jAER providing new types of events and sharing the same timestamping mechanism for all the possible events coming from the sensors or from event-based hardware algorithms.

References

[1] Posch, C.; Serrano-Gotarredona, T.; Linares-Barranco, B.; Delbruck, T., "Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output," Proceedings of the IEEE , vol.102, no.10, pp.1470,1484, Oct. 2014.

[2] Sivilotti, M., “Wiring Considerations in analog VLSI Systems with Application to Field-Programmable Networks”, Ph.D. Thesis, California Institute of Technology, Pasadena CA, 1991.

[3] Boahen, K.A. “Communicating Neuronal Ensembles between Neuromorphic Chips”. Neuromorphic Systems. Kluwer, Boston 1998.

[4] Chan, V.; Liu, S.-C.; van Schaik, A., "AER EAR: A Matched Silicon Cochlea Pair With Address Event Representation Interface," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.54, no.1, pp.48,59, Jan. 2007

[5] Koickal, T.J. et al. "Analog VLSI Circuit Implementation of an Adaptive Neuromorphic Olfaction Chip," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.54, no.1, pp.60,73, Jan. 2007

[6] Zamarreno-Ramos, C. et al., "Multicasting Mesh AER: A Scalable Assembly Approach for Reconfigurable Neuromorphic Structured AER Systems. Application to ConvNets," Biomedical Circuits and Systems, IEEE Transactions on , vol.7, no.1, pp.82,102, Feb. 2013

[7] Neil, D.; Liu, S.-C., "MINITAUR, an Event-Driven FPGA-Based Spiking Network Accelerator," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on. (in press and available online in 2014)

[8] jAER opensource project: http://sourceforge.net/p/jaer/wiki/Home/

[9] Liu, H.; Brandli, C.; Li, C.; Liu, S.-C.; Delbruck, T. “Design of a Spatiotemporal Correlation Filter for Event-based Sensors”. ISCAS 2015

[10]Gómez-Rodríguez, F. et al. “AER tools for Communications and Debugging”. Proc. IEEE ISCAS06. Kos, Greece May 2006.

[11]Brandli, C.; Berner, R.; Yang, M.; Liu, S.-C.; Delbruck, T. “A 240x180 130 dB 3us Latency Global Shutter Spatiotemporal Vision Sensor,” IEEE Journal of Solid-State Circuits, vol. Early Access Online, 2014.

[12]Delbruck, T.; Villanueva, V.; Longinotti, L., "Integration of dynamic vision sensor with inertial measurement unit for electronically stabilized event-based vision," ISCAS 2014, pp.2636,2639. Melbourne.

[13]Serrano-Gotarredona, R. et al., “CAVIAR: A 45k Neuron, 5M Synapse, 12GConnects/s AER Hardware Sensory–Processing–Learning-Actuating System for High-Speed Visual Object Recognition and Tracking,” IEEE Trasns. on Neural Networks, vol. 20, N. 9, pp. 1417- 1438, Sept. 2009.

[14]Berner, R.; Delbruck, T.; Civit-Balcells, A.; Linares-Barranco, A., "A 5 Meps $100 USB2.0 Address-Event Monitor-Sequencer Interface". ISCAS 2007. pp.2451,2454. New Orleans, USA.