Real time multiple objects tracking based on a bio-inspired processing cascade architecture

F. Gómez- Rodríguez, L. Miró-Amarante, F. Diaz-del-Rio, A. Linares-Barranco, G. Jimenez.
Robotics and Computer's Technology Group
University of Seville
Seville, Spain
gomezroz@us.es

***Abstract*— This paper presents a cascade architecture for bio-inspired information processing. We use AER (Address Event Representation) for transmitting and processing visual information provided by an asynchronous temporal contrast silicon retina. Using this architecture, we also present a multiple objects tracking algorithm; this algorithm is described in VHDL and implemented in a FPGA (Spartan II), which is part of the USB-AER platform developed by some of the authors.**

I. Introduction

This paper presents a cascade architecture for bio-inspired information processing; and an object tracking algorithm using this architecture. For visual information, provided by a silicon retina, processing and transmitting Address Event Representation is used; this use of events makes the system bio-inspired.

This object tracking algorithm has some differences from previous works in the field of objects tracking based on event processing. In [8], Delbruck et al. present an hybrid neuromorphic-procedural system for objects tracking, where a computer is used for events processing. In [9], Litzenberger et al. present an objects tracking system too, but now, a DSP is used for events processing. These two approaches need to collect events and render them like a video frame. Here only two events are needed to obtain the object center of mass, and only eight centers of mass are needed to obtain the object velocity. The algorithm is described in VHDL and implemented in a FPGA, so it is a fully hardware system.

Following we present the needed elements for developing and implementing the architecture and the object tracking algorithm: a) the *Address Event Representation* (AER) communication protocol[1][2]; b) the sensor, an asynchronous temporal contrast silicon retina, developed by the Institute of Neuroinformatics (INI) of the University of Zurich[3][4]; and c) the *USB-AER Board* [5][6][7], developed by the Robotics and Computer's Architecture Research Group.

A. The AER communication protocol

The AER protocol was proposed for neuro-inspired information transmitting from one neuro-inspired chip to another. The basics of AER consist of assigning an address to each cell (neuron) in a chip. Each cell transmits its activity showing its address in a common bus; two flow control signals are commonly needed (REQ and ACK), to start and stop the transmission.

As a result, the activity of every cell will appear in the common bus. Usually the activity is frequency coded. In this way, if cell's activity is high, its address will appear in the bus more frequently than other with lower activity. Each address occurrence, in the bus, is known as an *event.* So, we can say that the activity of each cell is coded in events frequency. Due to AER bus multiplexes all the events in a common bus, an arbiter is needed in the transmitter. Fig. 1 shows the organization of AER communication.

To transmit each event, a simple handshake protocol is normally used (see Fig. 2).

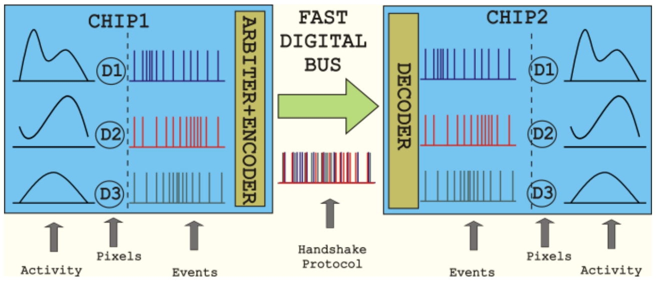

Figure 1. AER transmission organization. An address is asigned to each cell. The cell activity is transmited showing its address in a common bus. Each address ocurrence in the bus is known as an *event.* The cell activity is coded in event frequency.

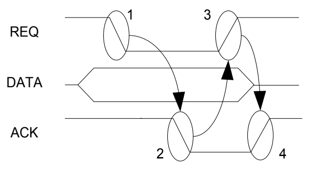

Figure 2. AER Handshake protocol

B. *The asynchronous temporal contrast silicon retina*

The sensor used in this work is a TMPDIFF128 silicon retina developed by Tobias Delbruck and his group at INI of University of Zurich [4]. Concretely, is an asynchronous temporal contrast silicon retina, with an AER interface.

Retina output consists of the pixels activity that corresponds to the scene movement. Fig. 3 shows how the retina output is. On the right a 2D histogram of the collected events for a period of time is showed. Events are signed to distinguish between positive or negative contrast changes in the pixel. Those pixels without changes in time do not produce events, so they don't appear in the histogram.

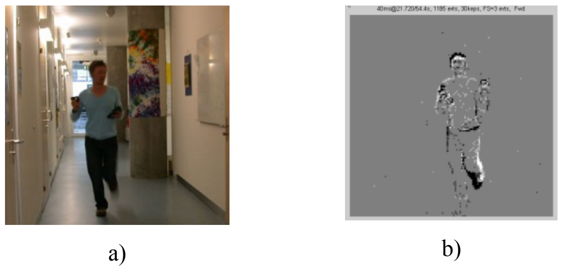

Figure 3. Retina functionality: a) Scene, b) Retina Output (These pictures have been extracted from [3])

C. *USB-AER Board*

The USB-AER board was developed by the Robotic and Computer's Technology Research Group of University of Seville, for giving support to the AER based systems developers. The first idea was to develop a board for AER-based systems testing and debugging, this board implements several modules for AER stream sequencing, monitoring, mapping (to change *on the fly* the address space), depending on the FPGA firmware.

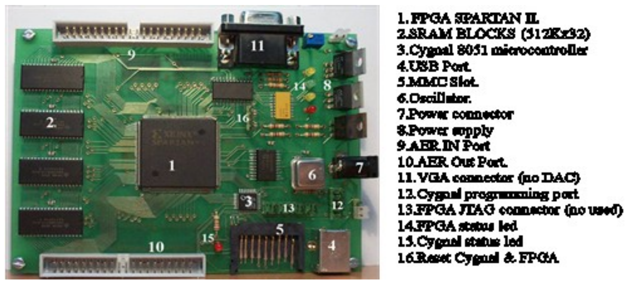

Figure 4. USB-AER Board

But the flexibility of USB-AER board's design makes possible to implement on it any other functionalities such as the presented one in this paper.

A picture of the USB-AER board is shown in Fig. 4. USB-AER board is based around a Spartan-II 200 Xilinx FPGA, with a 12ns, 512K by 32 bits SRAM memory bank. The board uses a Silicon Laboratories C8051F320 microcontroller to implement the USB and the MMC/SD interface.

II. Cascade Architecture for Bio-Inspired Information Processing

The proposed architecture consists of several cells, which extract, collect and process events from the AER stream, as soon as they are received. Each cell extracts and collects events depending on the application. Each one only retains the necessary events; the rest of events will pass through to the next cell.

The key point of this architecture is that events are processed as soon as they are received, without frame integration. Therefore the respond time of each cell is very short, in fact it is the delay time (order of ns).

Each cell *computes* the collected events and gives an output, which is an AER stream too. So, each cell has one AER input and two AER outputs, the first output gives the result of events *computation* and the second one resends the refused events. Fig. 5 shows the architecture overview.

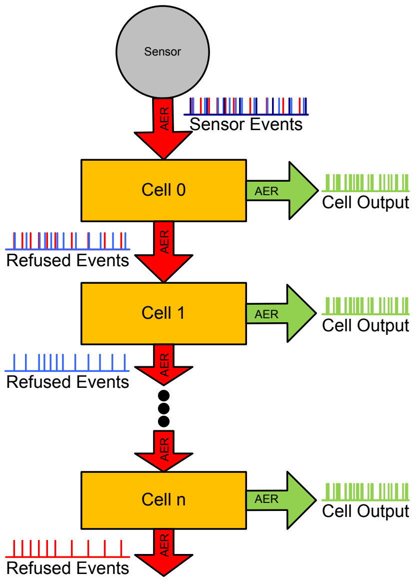

Figure 5. Architecture for bio-inspired information processing overview: Depending on the application, each cell only extracts and collects the necessary events from an AER stream, and the rest of the events are refused and resent to the next cell.

With this cascade architecture, the AER stream complexity is reduced from one cell to the next, because each cell does not resend the events used for *computing* its output. Furthermore, the cell output is also transmitted using AER. So, it can be used for feeding other *computing* layers. As a result, a bio-inspired parallel and multi-layer computing system could be obtained.

III. Multiple Objects Tracking Algorithm

In this section, we present an application using the architecture explained above. The application computes the position and velocity of several moving object in a scene. Obviously, objects have to be in movement, because the sensor (silicon retina) only *sees* the movement. So, the system's input is an AER stream that corresponds to the scene's movement.

For this application, each cell (of the architecture) consists of two sub-cells. One is programmed to *compute* the object center of mass, called CMCell; and its output is connected to other sub-cell that calculates the object velocity, called VCell (See Fig. 6)

CMCell collects events and calculates the object's center of mass. After a reset, CMCell is empty and takes the first event that it receives; after this first event CMcell reduces its field of view to a few pixels around center of mass, which coincides at the first with this initial event. If after a configurable time CMCell does not receive any event (in its field of view), CMCell will reset, and then it is supposed that the object is not moving any more. In opposite, if CMCell receives a second event (in itself field of view), CMCell computes the center of mass as the mean between both events position and the center of the field of view is updated with the new center of mass to track the object efficiently. Fig. 7 shows the CMCell state machine diagram.

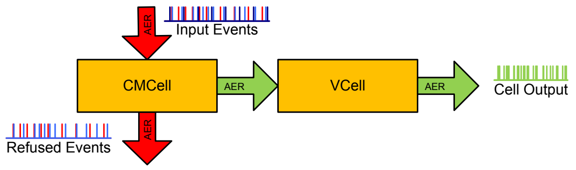

Figure 6. Cells for multiple objects tracking: CMCell calculates the object's center of mass, while VCell computes the object's velocity, eventually transmitted using AER.

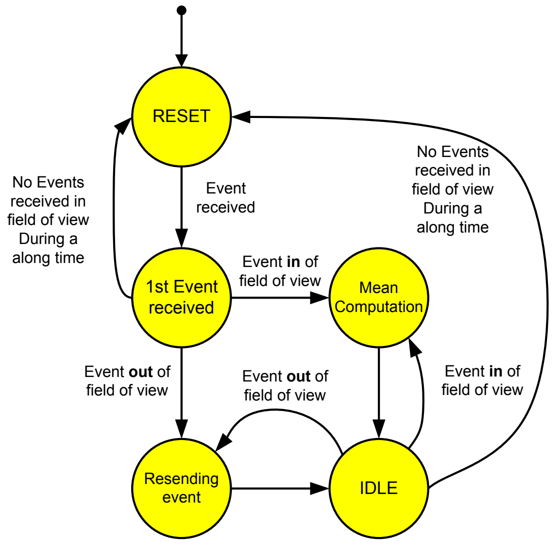
RESET
Event received
1st Event received
Event in of field of view
Mean Computation
No Events received in field of view During a along time
No Events received in field of view During a along time
Event out of field of view
Event out of field of view
Event in of field of view
Resending event
IDLE

Figure 7. CMCell state machine diagram

Using the center of mass obtained by CMCell, VCell calculates the velocity. For doing this, VCell stores a number of consecutives centers of mass, periodically distributed. We have determinate that with 8 values of center of mass is accurate enough. Moreover, the time between two center of mass values is configurable to adapt it to the velocity of the objects. The velocity is measured in the *image plane*, so it is necessary to know the distance between objects and the retina to calculate the real velocity.

As a result, the output of the system is an AER stream with the velocity of the objects.

This system has been described in VHDL and synthesized to configure the FGPA of the USB-AER board. Due to the reduce amount of gates of Spartan 2 (which is the USB-AER board's FPGA) only 6 objects can be tracked in parallel.

IV. EXPERIMENTS

The experiments consist of a silicon retina connected to an USB-AER board, and this last one connected to a computer.

In order to visualize the system output, every data from the VCell are stored in an internal memory, and then this data are downloaded to a PC and drawn using the Matlab function *quiver*.

In the following sections, we present two experiments.

A. *Four moving objects*

In this experiment the stimulus is a scene where there are 4 moving objects with different trajectories, but with constant velocities. Fig. 8 shows the experiment stimulus and the silicon retina output.

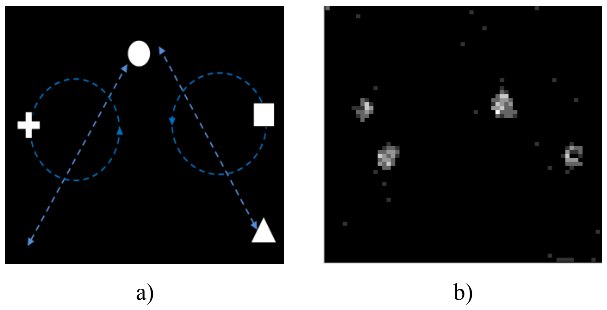

Figure 8. First experiment: four moving objects experiment. a) shows the stimulus composed by 4 objects (a cross, a circle, a triangle and a square) moving at the same velocity but with different trajectories; And b) shows a 64x64snapshot subsampled silicon retina output.

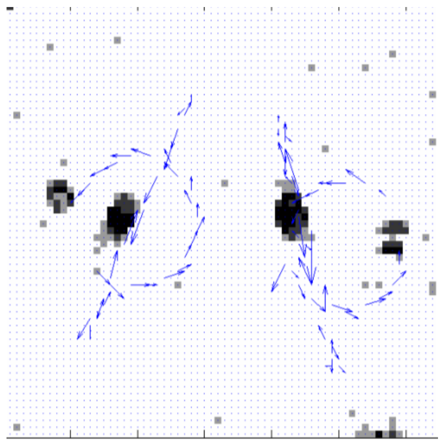

Figure 9. First experiment system output fused with silicon retina output. Arrows respresent the apparent object's velocity on the *image's plane*.

Fig. 9 shows the silicon retina output and the velocity vector for each moving objects. Velocity is expressed in unit of pixel per 20 milliseconds (which was the period of the VCells for collecting input events). In order to improve the quality of the result visualization, the retina output is drawn in negative. It can be observed that the system output (arrows) is quite similar to the objects trajectories. All velocity vectors modulus are similar, thus it is correct because the object's velocity is constant during the experiment (except for the circle and triangle objects, which velocity vectors modulus changes at the end of their trajectories, of course).

B. *One object moving at different velocities*

In this experiment the stimulus consists of one circle moving in the scene at different velocities. Circle starts moving slowly from the right bottom corner to left bottom corner, then moves faster to left up corner, then much faster to right up corner and then fastest to right bottom corner again. Fig. 10 shows the scene and the moving objects.

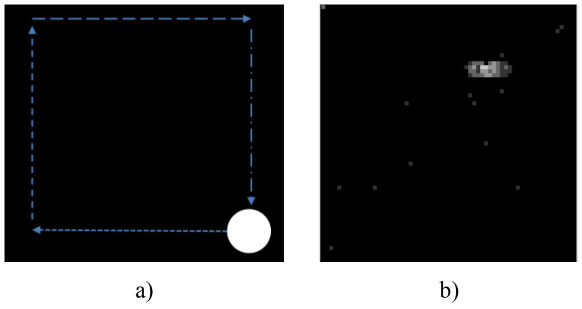

Figure 10. Second experiment: a) one circle moving at different velocity. Circle starts moving slowly from the right bottom corner to left bottom corner, then moves faster to left up corner, then much faster to right up corner and then fastest to right bottom corner again; And b) shows a 64x64snapshot subsampled silicon retina output

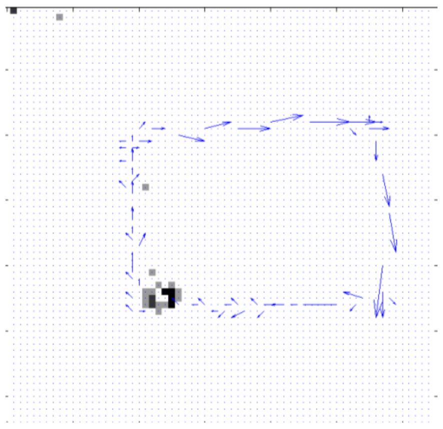

Figure 11. Second experiment system output fused with silicon retina output. Rows respresent the apparent object's velocity on the *image's plane*.

Fig. 11 shows the result of this second experiment; the retina output is again in negative to improve the quality of the visualization. It can be observed that the system can detect different velocities of the object. In this experiment the velocity *computation* has some limitation, for example, in the second part of the slowest trajectory (left part of the figure). This is because the system tries to give a respond as soon as possible; and because the period of VCell is very small related to the object's velocity.

V. Conclusion

This paper presents a bio-inspired processing cascade architecture, which is successfully used for implementing a multiple objects tracking algorithm. The system is fully hardware implemented, described in VHDL; each CMCell+VCell block only requires 161+182 slices of a Spartan-II 200 Xilinx FPGA.

Results show that it is possible to estimate several objects' velocities *on the fly*, that is, as soon as events are received; without frame integration. Therefore obviously, estimation has some limitations mainly when the object's velocity is less the respond time, but this can be improved by increasing the system respond time (only 20 ms in these experiments). It is fully scalable to track an arbitrary number of objects.

To enlarge the velocity interval range where the system velocity estimation is poor, it is necessary a higher level layer devoted to detect and correct *errors* in the velocity estimation. Nevertheless the required quality of the velocity estimation depends on the high level application that will use the velocities.

Acknowledgment

Authors would like to thank to Tobias Delbruck and his group for their silicon retina. Authors also thank to Spanish grants Andalusian Government Excellence Research project P06-TIC-02298, VULCANO (TEC2009-10639-C04-02) and BrainSystems (P06-TIC-01417) for supporting this work.

References

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

[2] Kwabena A. Boahen. "Communicating Neuronal Ensembles between Neuromorphic Chips". *Neuromorphic Systems*. Kluwer Academic Publishers, Boston 1998.

[3] P. Lichtsteiner, et al., "A 128×128 120dB 30mW Asynchronous Vision Sensor that Responds to Relative Intensity Change," ISSCC Dig. of Tech. Papers, San Francisco, 2006, pp. 508-509 (27.9).

[4] Lichtsteiner, Delbruck, Posch: "A 128x128 120 dB 15 µs Latency Asynchronous Temporal Contrast Vision Sensor". , Solid-State Circuits, IEEE Journal of, ISSN 0018-9200; vol.43, no.2, Feb. 2008, p. 566-576

[5] F. Gómez-Rodríguez, R. Paz, A. Linares-Barranco, M. Rivas, L. Miró, G. Jiménez, A. Civit. "AER tools for Communications and Debugging". *Proc. IEEE ISCAS06.* Kos, Greece, May 2006.

[6] M. Rivas, F. Gomez-Rodriguez, R.Paz, A. Linares-Barranco, S. Diaz, D. Cascado. "Tools for Address-Event-Representation Comunication Systems and Debugging". Lecture Notes in Computer Science. Vol. 696. 2005. Pag. 289-296

[7] R. Paz, F .Gomez-Rodriguez, M.A. Rodriguez, A.Linares-Barranco, G.Jimenez, A. Civit. "Test Infrastructure for Address-Event-Representation Communications". Lecture Notes in Computer Science. Vol. 3512. 2005. Pag. 518-526

[8] T. Delbruck, T. and P. Lichtsteiner . "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system". ISCAS 2007, New Orleans, 27-30 May 2007 Pag:845 - 848.

[9] Litzenberger, M.; Bauer, D.; Belbachir, A.N.; Garn, H.; Kohn, B.; Posch, C.; Schön, P.; "Embedded Vision System for Real-Time Object Tracking Using an Asynchronous Transient Vision Sensor"; 12th IEEE Workshop on Digital Signal Processing and Signal Processing Education DSP/SPE 2006; Wyoming, USA;; ISBN: 1-4244-0535-1; p. 173-178; September, 24-27, 2006.