Synthetic Generation of Address-Events for Real-Time Image Processing

A. Linares-Barranco, R. Senhadji-Navarro, I. García-Vargas, F. Gómez-Rodríguez, G. Jimenez and A. Civit.
Arquitectura y Tecnología de Computadores.
Universidad de Sevilla.
Av. Reina Mercedes s/n, 41012-Sevilla
SPAIN

Abstract – **Address-Event-Representation (AER) is a communication protocol that emulates the nervous system's neurons communication, and that is typically used for transferring images between chips. It was originally developed for bio-inspired and real-time image processing systems. Such systems may consist of a complicated hierarchical structure with many chips that transmit images among them in real time, while performing some processing (for example, convolutions). In this paper several software methods for generating AER streams from images stored in a computer's memory are presented. A hardware version will work in real-time. All of them have been evaluated and compared.**

I. INTRODUCTION

Address-Event-Representation (AER) was proposed in 1991 by Sivilotti [5] for transferring the state of an array of analog time dependant values from one chip to another. It uses mixed analog and digital principles and exploits pulse density modulation for coding information. Fig. 1 explains the principle behind the AER basics.

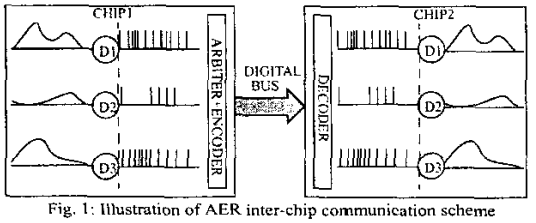

Fig. 1: Illustration of AER inter-chip communication scheme

The Emitter chip contains an array of cells (like, for example, a camera or artificial retina chip) where each pixel shows a continuously varying time dependant state that changes with a slow time constant (in the magnitude order of milliseconds). Each cell or pixel includes a local oscillator (VCO) that generates digital pulses of minimum width (a few nanoseconds). The density of pulses is proportional to the state of the pixel (or pixel intensity). Each time a pixel generates a pulse (which is called "event"), it communicates to the array periphery and a digital word representing a code or address for that pixel is placed on the external inter-chip digital bus (the AER bus). Additional handshaking lines (Acknowledge and Request) are also used for completing the asynchronous communication. The inter-chip AER bus operates at the maximum possible speed. In the receiver chip the pulses are directed to the pixels or cells whose code or address was on the bus. This way, pixels with the same code or address in the emitter and receiver chips will "see" the same pulse stream. The receiver pixel integrates the pulses and reconstructs the original low frequency continuous-time waveform, like a neuron does in the nervous system. Pixels that are more active are accessing the bus more frequently than those less active.

Transmitting the pixel addresses allows performing extra operations on the images while they travel from one chip to another. For example, inserting properly coded memories (ie. EEPROM) allows transformation (ie. shifting and rotation) of images. Also, the image transmitted by one chip can be received by many receiver chips in parallel, by properly handling the asynchronous communication protocol. The peculiar nature of the AER protocol also allows for very efficient convolution operations within a receiver chip [8].

There is a growing community of AER protocol users for bio-inspired applications in vision and audition systems, as demonstrated by the success in the last years of the AER group at the Neuromorphic Engineering Workshop series [1]. The goal of this community is to build large multi-chip and multi-layer hierarchically structured systems capable of performing complicated array data processing in real time. The success of such systems will strongly depend on the availability of robust and efficient development and debugging AER-tools. One such tool is a computer interface that allows not only reading an AER stream into a computer and displaying it on its screen in real-time, but also the opposite: from images available in the computer's memory, generate a synthetic AER stream in a similar manner as would do a dedicated VLSI AER emitter chip [2][4][5][6].

This work is involve in the framework of the European Research project CAVIAR, one of the objectives of CAVIAR is develop an AER-computer interface. This paper presents several methods for synthetic AER streams generation (section II). It also evaluates these methods attending to the execution time comparison and to the error of distribution of the events along the time associated to an image: distance to the frequency of an intensity level. The methods are evaluated for images with different average intensity levels with a Gaussian histogram (from 10-90% charge of events) and for a typical image (around 50 % charge of events) (section III). Conclusions presented in this paper are not definitive. A hardware implementation and its study is necessary for a real-time comparison.

II. SYNTHETIC STREAM GENERATION

There are many software algorithms to transform a bitmap image into an AER stream of pixel addresses. In all

of them the frequency of appearance of the address of a given pixel must be proportional to the intensity of that pixel. Note that the precise location of the address pulses is not critical. The pulses can be slightly shifted from their nominal positions; the AER receivers will integrate them to recover the original pixel waveform.

Whatever algorithm is used, it will generate a vector of addresses that will be sent to an AER receiver chip via an AER bus. If we have an image of *NxM* pixels and each pixel can have a grey level value from 0 to *K*, one possibility is to place each pixel address in the address sequence as many times as the value of its intensity, and distributed in equidistant position. In the worst case (all pixels with value *K*), the address sequence would be filled with *NxMxK* addresses. The time used for transmitting an image needs to be the same for any image intensity charge, leaving blank slots if non-event has to be sent. Each algorithm would implement a particular way of distributing these address events. Let us propose some algorithms: the Scan method, the Scan Slice, the Uniform method, the Random method, the Random-Square method and the Exhaustive method.

A. The Scan method

In this method the image is scanned many times. For each scan, every time a non-zero pixel is reached its address is put on the address sequence in the first available slot, and the pixel value is decremented by one. This method is very fast, due to it does not need to look for empty slots, although the image needs to be scanned many times (*K* times in the worst case). However, a non-well event distribution is obtained due to all pulses of the pixels with low intensity values will appear only at the beginning of the sequence.

B. The Scan-Slice method

The Scan method can be enhanced in the distribution of events if an blank slot is left when all the events for a pixel have been sent. The image is scanned many times, as the Scan method does, but when a pixel's value is zero, a blank slot is left in the time period.

Both the Scan method and the Scan-Slice method can be easily implemented in hardware, because the time period is generated sequentially. So, the required hardware could be a memory for store the image, plus the control circuitry.

C. The Uniform method

In this method, the image is scanned pixel by pixel only once. For each pixel, the generated pulses must be distributed at equal distances. As the sequence is getting filled, the algorithm may want to place addresses in slots that are already occupied. This situation is called 'collision' and it appears with the AER approximation of inter-chip communication. In this case, it will put the pulse in the nearest empty slot of the time period. So, this method, apparently, will make more mistakes at the end of the process than the beginning. The execution time grows considerably because the collisions have a high-cost in time to resolve it.

This method can be enhanced by applying a small shift before placing the events in the time period. This shift can be the address of the pixel. For example, the pixel (i,j)=(10,10) will be shifted 1290 positions respect to the beginning of the time period (*N=M=128*). These positions are translated as a delay in the transmission of the event.

The hardware implementation of this method is under study. But, the time period is not generated sequentially, so, the first implementation seems to need a memory to save the complete time period before transmitting it.

D. The Random method

This method places the address events in the slot positions obtained by a pseudo-random number generator based on Linear Feedback Shift Registers (LFSR) [7][9]. Due to the properties of the LFSR used, each slot position is generated only once and no collisions appear. If a pixel in the image has the intensity *p*, then the method will take *p* values from the pseudo-random number generator and places the pixel address in the corresponding *p* slots of the address sequence. They will not be equidistant but will appear along the complete address sequence. This method is fast, because the image is swept only once, and because the algorithm does not need to perform searches for empty slots.

Due to the LFSR can obtain two consecutive, or very closer, addresses in a few calls, The LFSR-based method can be enhanced using a *b*-bit counter for the high part of the address. So, for each call to the address generator, 2^b addresses equally distributed are obtained. For pixels with a grey level value near to 2^b, the distribution obtained is similar to the ideal distribution. However, for different grey level, the distribution of the events gets worse again. Tuning the correct size counter value for each image, the random method could be improved.

Fig. 2 shows the LFSR structure with a 2-bit counter for a 128x128 images with a 256 grey levels.

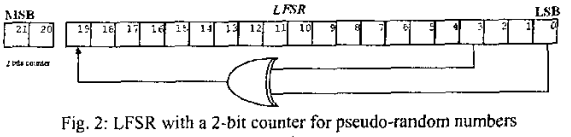

Fig. 2: LFSR with a 2-bit counter for pseudo-random numbers generation.

E. The Random-Square method

Using the Random method with a fixed size counter from 1 to the maximum grey level, the event distribution for high level pixels is acceptable, but poor for low level values. Substituting the counter by another LFSR, the distribution could be improved.

For a 128x128 image with maximum grey level of 255, 8-bit LFSR (LFSR-8) is used for selecting 255 slices of 128x128 positions, and another 14-bit LFSR (LFSR-14) selects the position inside the slice. The image is scanned only once. For each pixel a 14-bit number is generated by the LFSR-14, and the LFSR-8 is called as many times as the intensity level of the pixel would indicate. Fig. 3 shows the LFSRs used by the Random-Square method.

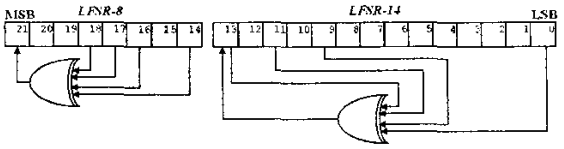

Fig. 3: LFSR-8 and LFSR-14 used by the Random-Square method.

F. *The Exhaustive method*

This algorithm also divides the address event sequence in K slices of NxM positions for an image of NxM pixels with a maximum grey level of K. This method is based on the idea that each possible event (as maximum $NxMxK$) has assigned one fixed position in the address event sequence. In such way, for the slice k, an event of the pixel (i,j) is sent on the time t if the following condition is asserted:

$$(k \cdot P_{i,j}) \bmod K + P_{i,j} \geq K \quad \text{and}$$
$$N \cdot M \cdot (k-1) + (i-1) \cdot M + j = t$$

where $P_{i,j}$ is the intensity value of the pixel (i,j) (see Fig. 4).

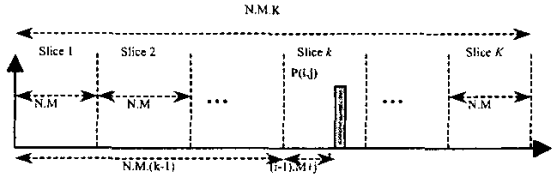

Fig. 4: The event distribution made by the Exhaustive method.

The Exhaustive method tries to distribute the events of each pixel into the K slices at equal distances. For this propose, the algorithm scan the image K times. In the iteration k, if the previous condition is true, then the corresponding event is sent, otherwise the algorithm will wait for the following event (no event is sent in time t).

III. SIMULATION RESULTS

This section is devoted to compare the methods proposed above and to estimate how the performance of the methods is affected by the type of image. To carry out this analysis a set of random images were generated, which represent a small population of images. In others population of images, with different histogram patterns and charge of events, the results can vary in some methods, although the behaviour will be closer to the results here presented.

Theses images was obtained considering two aspects: its histogram must be closer to Gaussian distribution and the number of events needed to transmit them, this is called "image event charge" and; this way, 100% event charge corresponds to an image with all of pixel with maximum value. In such a way, a image with 10% of event charge, represent an image that used 10% of the possible event. Fig. 5 shows the images used.

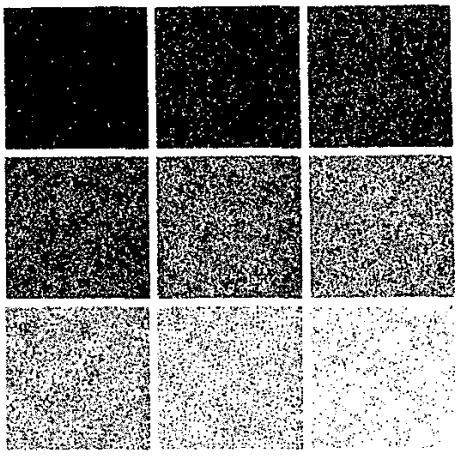

Fig. 5: Test image set (10% to 90% event charge)

All of the methods have been implemented in C++ language and an application has been developed which allows the generation of the address events from a grey-scale image and calculate the parameters previously described. The Fig. 6 shows the screenshot of this software interface.

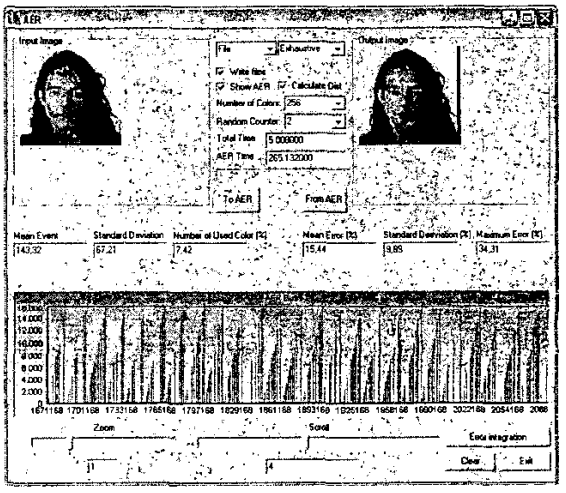

Fig. 6: Software interface

A. *The execution time*

The interest of this point is not to compare the execution time of each method, because the methods have to be implemented in hardware for a more real comparison. Therefore the interest yields in the behaviour of the execution time for each method respect to the image event charge. The test has been made by using the nine images showed in Fig. 5.

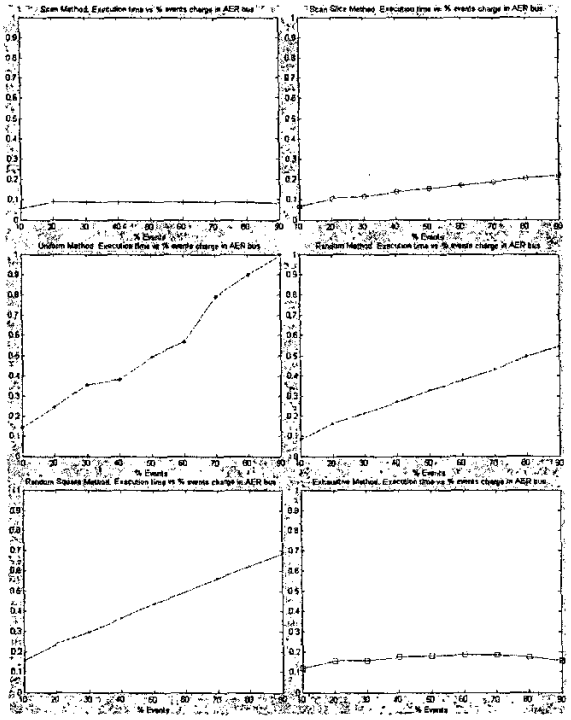

Fig. 7: Execution time for Scan, Uniform, Random, Random Square, Scan Slice and Exhaustive method.

Fig. 7 shows the execution time versus the charge of events in the image. Scan and Exhaustive methods follow an almost constant relation because the charge of events is traduced into an insignificant increasing in time execution, although this affect to the Exhaustive method in images with half charge of events. Random, Random Square and Scan Slice methods follow a growing up behaviour, due to the increasing number of events, with a less effect in the Scan Slice. Uniform method is also affected by collisions (grows up faster than others), but it seems to be a relation between the number of collisions and the histogram, as can be seen comparing theses results with the typical image.

B. *The distribution error*

In an ideal AER distribution all events for one pixel and image could be equidistant in time: constant frequency of events. In this section, the distribution of event obtained with each method is evaluated. The distribution error proposed measures how much the event distribution generated by a method deviates from the ideal distribution.

Let suppose D_{ij} is the ideal distance between events of the pixel *(i,j)* of a *NxM* image with *K* grey level values.

$$D_{i,j} = \frac{N \cdot M \cdot K}{P_{i,j}}$$

where $P_{i,j}$ is the intensity value of the pixel *(i,j)*.

Let suppose $d_{i,j}^k$ is the distance between the *k*-event ($p_{i,j}^k$) and the following event ($p_{i,j}^{k+1}$). The distance of the last event is calculated supposing that the next event is the first of a new sequence of the same image (we suppose that the image is continuously re-sent).

$$d_{i,j}^k = p_{i,j}^{k+1} - p_{i,j}^k$$

Then we can measure the mean error for a pixel as the average of the differences between the ideal and real distance. The error expression is:

$$e_{i,j} = \frac{\sum_{k=1}^{P_{i,j}} \left| D_{i,j} - d_{i,j}^k \right|}{P_{i,j}}$$

It is easy to see that the worst case for this error measurement is which all the events are together in the address sequence (see Fig. 8).

Therefore, in order to compare the error obtained for different methods and images, the error of each pixel must be normalized respect to the maximum error associated to the pixel. The following expression is the maximum error for the pixel *(i,j)*:

$$me_{i,j} = 2 \cdot (D_{i,j} - 1) \cdot (1 - \frac{1}{P_{i,j}}) \text{ with } P_{i,j} \neq 1$$

For $P_{i,j} = 1$, the distribution error is zero, due to only one event has to be sent.

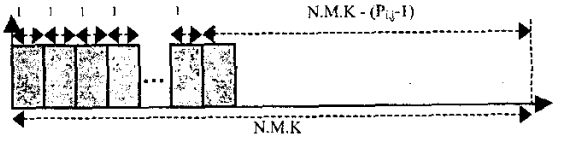

Fig. 8: The worst event distribution.

Finally, we define a matrix with the same dimensions of the image, where the *(i,j)* element represents the error normalized for the pixel *(i,j)*:

$$NE = \begin{pmatrix} e_{1,1}/me_{1,1} & e_{1,2}/me_{1,2} & \cdots & e_{1,M}/me_{1,M} \\ e_{2,1}/me_{2,1} & e_{2,2}/me_{2,2} & \cdots & e_{2,128}/me_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ e_{N,1}/me_{N,1} & e_{N,2}/me_{N,2} & \cdots & e_{N,M}/me_{N,M} \end{pmatrix}$$

Fig. 9 shows the normalized distribution error calculated for the nine test images using the methods proposed. The x-axis represents the image event charge and the y-axis is the error.

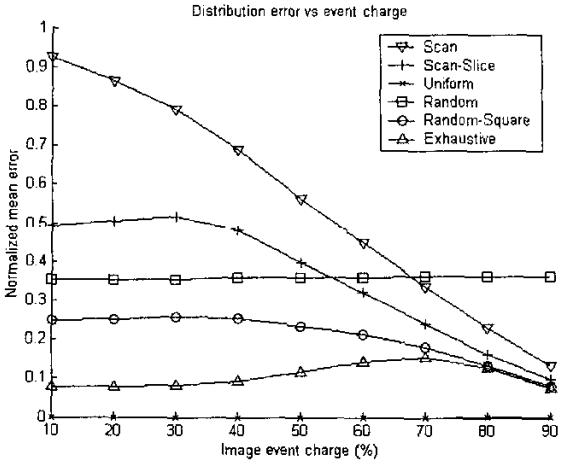

Fig. 9: Distribution error

As it can be seen, the minimum error is obtained with the exhaustive method.

The error decreases with the image event charge in all methods except Random. In the scan methods the pixels with low intensity has a higher error of distribution inside the time period selected for transmitting the image.

The Uniform method has the best results for the population of image selected, because there are very few collisions. This method represent the closer solution to the ideal distribution in the context presented.

Let us consider a typical case, Fig. 10 shows a typical grey level image and Fig. 11 its histogram.

Fig. 10: Typical image

Table I shows the maximum and mean normalized errors in percent. As it can be seen, the Uniform method proposed yields a significant improvement over the other methods. It can be observed that the values for Uniform method correspond to an image between the 80 and 90 % of charge for the Gaussian histogram population showed in Fig. 5 Between the methods that divide the time period in K windows or slice, the Exhaustive method obtain the best results. The Random-Square results are better than those obtained by the Random method.

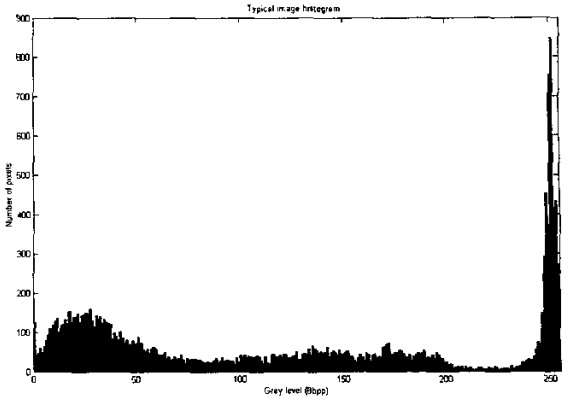

Fig. 11: Typical image histogram

Table 1: Normalized errors for the methods proposed.

Method	Mean	Std. Deviation	Max.
Scan	69.75	17.14	99.22
Scan-Slice	49.19	35.58	99.22
Uniform	0	0.01	0.28
Random (2bits)	34.92	8.92	84.95
Random-Square	22.74	14.69	98.43
Exhaustive	4.91	5.22	17.25

IV. CONCLUSIONS AND FUTURE WORKS

In this paper six software methods for generating AER streams from images stored in a computer's memory have been presented. The different methods have been analysed and tested by simulation software. It is necessary the hardware implementation for real-time, and the corresponding study.

A hardware platform that exploits these techniques is currently under development. For this goal, an analysis of different architectures for real time is being made. A dedicated hardware to write and read to/from an AER bus is being developed using a standard FPGA-based prototyping board.

It is still soon to conclude that the Uniform method is the most adequate method, as it can be expected. The hardware study will complement the work presented in this paper.

For more specific populations of images, the methods need to be analysed to clarify the most appropriate. There are several factors that can determine the selection of the method for the desired population: the necessity of real-time, the distribution of events, whose have been analysed here. But it can be interesting to analyse the effect of the sequence of events in traffic effect in the AER bus, where the density of events in time could make worse the results of the methods. For example, in biomedical applications, the x-ray or ultrasound-scan images have a non-gaussian histogram.

V. ACKNOWLEDGMENTS

The authors wish to express their gratitude to the support given to this work by the European Commission through project CAVIAR "Convolution AER Vision Architecture for Real-Time" (IST-2001-341024), funded by the European Commission, V Framework Programme "Information Society Technologies (IST) Programme".

VI. REFERENCES

[1] A. Cohen, R. Douglas, C. Koch, T. Sejnowski, S. Shamma, T. Horiuchi, and G. Indiveri, *Report to the National Science Foundation: Workshop on Neuromorphic Engineering*, Telluride, Colorado, USA, June-July 2001. [www.ini.unizh.ch/telluride]

[2] A. Linares-Barranco. *"Estudio y evaluación de interfaces para la conexión de sistemas neuromórficos mediante Address-Event-Representation"*. Ph.D. Thesis, University of Seville, Spain, 2003

[3] Charles M. Higgins and Christof Koch. *Multi-Chip Neuromorphic Motion Processing*. January 1999.

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

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

[6] Misha Mahowald. *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function*. Ph.D. Thesis. California Institute of Technology Pasadena, California 1992.

[7] Pierre L'Ecuyer, François Panneton. *A New Class of Linear Feedback Shift Register Generators*. Proceedings of the 2000 Winter Simulation Conference.

[8] Teresa Serrano-Gotarredona, Andreas G. Andreou, Bernabé Linares-Barranco. *AER Image Filtering Architecture for Vision-Processing Systems*. IEEE Transactions on Circuits and Systems. Fundamental Theory and Applications, Vol. 46, N0. 9, September 1999.

[9] Linear Feedback Shift Register V2.0. Xilinx Inc. October 4, 2001. http://www.xilinx.com/ipcenter.