Accelerating Motion Analysis Algorithms with a Pixel Change-Driven Scheme

F. Vegara[1] J.A. Boluda[1] J. Domingo[2] F. Pardo[1] X. Benavent[2]
[1]Department d'Informàtica, Universitat de València, Burjassot, Spain
[2]Institut de Robòtica, Universitat de València, Paterna, Spain

Abstract— *Motion analysis is a computationally demanding task due to the large amount of data involved and to real-time requirements. On the other hand, biological visual schemes are an interesting source for the improvement of artificial visual systems. In this paper, a biologically inspired strategy based on delivering and processing pixels, instead of processing the complete frame, is presented. Only the pixels of interest in each frame are processed, taking as that those which have suffered higher changes. As an example, two applications are shown: a tracking algorithm and the Horn and Schunck optical flow algorithm. Results show that the implementations of this strategy in both algorithms are simple, achieving an execution time speed-up while keeping results similar to classical approaches.*

Keywords: Motion analysis, Tracking algorithms, Optical flow

1. Introduction

Motion analysis is an important issue in artificial vision. Implementations of these algorithms with real-time restrictions are difficult due to the high computational cost of the algorithms employed and the large amount of data to be processed. This limitation has already been addressed from the viewpoint of selective reduction of information using space-variant sensors [1], [2]. Unfortunately, this approach is not always applicable. The use of these sensors requires sometimes a severe algorithm adaptation which often makes this idea infeasible for data reduction. Industry and academic sensor researchers have focused their efforts into increasing frame speed and accuracy of CCD or CMOS sensors.

Commonly, the classical approach for motion analysis is full image processing, where each image is a snapshot taken at regular intervals. The normal procedure implies the application of the processing algorithm for each image in the sequence. This approach is enough when the system is capable of detecting different moving objects in few milliseconds, but if the algorithm is complex, the processing time can last even seconds for each image in the sequence. Biological systems work in a different manner: each sensor cell sends its illumination information asynchronously [3]. In this way, it is also possible to implement a data-flow policy in the algorithm execution, processing only those pixels that have changed most. This strategy will decrease the total amount of data to be processed.

2. Selective Change-Driven camera and processing

Some visual sensors have been designed to take advantage of the selective reduction of information in a biologically inspired manner. In a similar way, the asynchronous nature of the biological visual systems has been already partially emulated [4]. In these artificial sensors the change event signaling depends on a contrast sensitivity threshold, which is also found in most biological vision systems. A pixel change greater than this threshold is considered as a change, and consequently this pixel is read out and processed. This strategy has been already utilized for speeding-up differential motion detection algorithms [5].

The advantages of the individual pixel reading for high-speed motion estimation of a Selective Change Driven (SCD) camera have been already presented [6]. In this camera every pixel works independently of the others. Every pixel has an analogue memory with the last read-out value. The absolute difference between the current and the stored value is compared for all pixels in the sensor; the pixel that differs most is selected and its illumination level and address are read out synchronously or asynchronously for processing. In this way, the pixels will be sent, and thus processed in a data-flow manner, ordered by its illumination change.

The goal is to know if this technique can accelerate motion analysis algorithms while keeping an accurate enough motion estimation. Two motion analysis algorithms have been adapted to the pixel change-driven strategy to answer this question.

3. A simple tracking algorithm

Tracking algorithms are a milestone in image sequence processing and their computational costs worth the trial of new implementations. Moreover, they are a very useful task in robot vision, particularly for biologically inspired robots.

This application uses change-driven processing to track one or several objects in real time; objects can be described by any means (border detection, textured areas, etc.) as long as the description can yield the coordinates in the image plane of a characteristic point of the object. The advantage on using a change-driven approach arises in those methods in which the calculations to find the characteristic point can be accelerated processing only the biggest change.

The goal is to track one or more mobile objects on a static background. In order to do this, objects are first segmented. A simple segmentation algorithm has been chosen which consists on thresholding of the gray level input image followed by the calculation of the centre of mass of each detected blob (set of connected pixels). It is important to point out that neither the number of elements nor the mechanism for object detection and separation have to be necessarily those used here. The complete thresholding, object separation and calculation of centers of mass is done only for the first frame. Later on, the values are updated using only the pixels whose gray level has changed significantly.

3.1 Change-Driven tracking algorithm

Every ΔT seconds the coordinates of the pixel that has changed most, as selected by the voting mechanism of the sensor, together with the value of its gray level variation, are received. Then, it is checked if that pixel corresponds to any of the objects being tracked, and if so, to which one. This is done by getting its distance to each centre of mass; if the distance is lower than the object size the pixel is assigned to the closest object. This implicitly assumes convex non-overlapping objects. Then, according to the gray level variation, the pixel is classified as formerly object/currently background or vice versa. Whatever the case, the gray level of the affected pixel is updated and the centre of mass of the involved object is recalculated.

The centre of mass calculation is very fast since it involves few operations: adding or subtracting the gray level of the changed pixel to the total sum and incrementing or decrementing the total number of pixels in the object, followed by division of both quantities.

3.1.1 Variable definition

- ΔT: time elapsed between two successive deliveries by the sensor (equivalent to the sampling time, even it has not to be uniform).
- (x_n, y_n): coordinates of the pixel which has changed most in the latest ΔT seconds, so it is said, the pixel sent by the sensor in a given moment. In the unlikely case two pixels have changed in exactly the same amount the sensors selects one according to predefined rules (for example, the top-most left) but this can be changed in each iteration and anyway the discarded pixels are likely to be selected for later deliveries.
- Δ_n: the gray level variation of the selected pixel. It is sent by the sensor together with the coordinates of such pixel.
- N^i: number of pixels that currently belong to object i
- k: number of objects to be tracked.
- $(\tilde{x}^i_{cm}, \tilde{y}^i_{cm})$: coordinates of the center of mass of object $i (i = 1...k)$.
- μ: threshold for determining the ascription of a pixel to any object, so that:
  if level $\geq \mu$ $\rightarrow$ pixel belongs to an object.
  if level $< \mu$ $\rightarrow$ pixel belongs to background.
- d_n: current gray level of pixel (x_n, y_n).
- R_i: minimum radius of circumference centered at the center of mass of object i that contains the object. This radius is affected by some small tolerance (ΔR) to adjust the decision process. This mechanism of decision could be changed by a more complex one to deal with the case of non-convex objects or overlapping.

3.1.2 Algorithm calculations

The algorithm is composed by a first step that thresholds the initial frame, separates the objects by surrounding each one's contour, determines the radius of each enclosing circumference and calculates the initial centre of mass. Then an endless loop driven by the arrival of new data from the camera every ΔT seconds runs as follows:

- Wait until (not_received_new_data)
- Receive (x_n, y_n, Δ_n).
- Calculate the index j of the object whose center of mass is at minimal distance of (x_n, y_n) as:

$$min_j \left[\left(x_n - \tilde{x}^j_{cm}\right)^2 + \left(y_n - \tilde{y}^j_{cm}\right)^2 \right] \quad (j \leq k) \quad (1)$$

  Let us call this index p.
- If (x_n, y_n) is closer to $(\tilde{x}^p_{cm}, \tilde{y}^p_{cm})$ than $R(p)$ make
  - $d_n = d_n + \Delta_n$
  - If the pixel was not a member of the object but it has become one, i.e. if

$$\{[(d_n + \Delta_n) \geq \mu] \text{ AND } (d_n < \mu)\} \quad (2)$$

    update number of pixels of object p as $N_p \rightarrow N_p + 1$ and recalculate its center of mass:

$$\tilde{x}^p_{cm} = \frac{N_p}{N_p + 1}\tilde{x}^p_{cm} + \frac{1}{N_p + 1}x_n \quad (3)$$

$$\tilde{y}^p_{cm} = \frac{N_p}{N_p + 1}\tilde{y}^p_{cm} + \frac{1}{N_p + 1}y_n \quad (4)$$

  - If the pixel was a member of the object but it is not now, i.e. if

$$\{[(d_n + \Delta_n) \leq \mu] \text{ AND } (d_n > \mu)\} \quad (5)$$

    update number of pixels of object p as $N_p \rightarrow N_p - 1$ and recalculate its center of mass again as shown in equations 3 and 4.
    The new center of mass is obviously a weighted mean of the old center and the coordinates of the new added pixel since

$$\tilde{x}_{cm}^{p} = \frac{1}{N^p}\sum_{i=1}^{N^p} x_i^p \tag{6}$$

so when a new pixel arrives

$$\frac{N^p\tilde{x}_{cm}^{p} + x_n}{N^p+1} = \frac{\left(\sum_{i=1}^{N^p} x_i^p\right) + x_n}{N^p+1} = \\ = \frac{1}{N^p+1}\sum_{i=1}^{N^p+1} x_i^p \tag{7}$$

and similarly for the y coordinate.

- In any other case the received pixel is considered noise and no action is taken.

4. Optical Flow computation

Optical flow can be defined as the apparent movement of the intensity level of an image. Unfortunately, the changes in intensity are not exclusively due to the real movement of the visible surfaces of the object present in the scene but also to changes in the illumination or to noise [7]. In these cases an apparent movement is observed which does not correspond to real displacements so optical flow is not always equal to movement field.

Nevertheless, optical flow is one of the main methods to estimate movement of objects in the 3D space and its calculation provides valuable information not only to artificial systems but also to biological ones.

Unfortunately its calculation is computationally intensive which uses to prevent or severely limit its use in real time applications; despite of this its high scientific interest motivates research on new strategies and techniques to reduce its calculation time [8], [9], [10], [11], [12], [13].

4.1 Horn and Schunk proposal

One of the most important contributions to the calculation of the optical flow was provided by Horn and Schunck [14]. In this work differential techniques (spatio-temporal gradient) are applied under global restrictions to solve the aperture problem. Several approximations are assumed, mainly the conservation of intensity (gray level in any given point is kept constant along time, except changes due to displacement, i.e. illumination changes are neglected). Under this assumption the following equation holds:

$$I(x,y,t) = I(x+u\delta t, y+v\delta t, t+\delta t) \tag{8}$$

where $I(x,y,t)$ is the image intensity at point (x,y) and time t and (u,v) are the horizontal and vertical components of the optical flow vector measured at (x,y).

Assuming also that image intensity changes in a smooth way spatially as long as temporally it can be developed in series and discard second and higher order terms which leads to

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{9}$$

The procedure used to calculate spatial and temporal gradients consists on approximating them by a convolution operation with a 8-neighbour mask that written as a formula results to be

$$\begin{aligned} I_x &= \frac{\partial I}{\partial x} = \frac{1}{4}(I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} + \\ &\quad + I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1}) \\ I_y &= \frac{\partial I}{\partial y} = \frac{1}{4}(I_{i+1,j,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i,j+1,k} + \\ &\quad + I_{i+1,j,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i,j+1,k+1}) \\ I_t &= \frac{\partial I}{\partial t} = \frac{1}{4}(I_{i,j,k+1} - I_{i,j,k} + I_{i+1,j,k+1} - I_{i+1,j,k} + \\ &\quad + I_{i,j+1,k+1} - I_{i,j+1,k} + I_{i+1,j+1,k+1} - I_{i+1,j+1,k}) \end{aligned} \tag{10}$$

where sub indexes (i,j,k) refer to pixel with spatial coordinates (i,j) at time k.

To eliminate the aperture problem an additional constraint is needed. Horn and Schunck introduce a method of global restriction that consists on minimizing the squared magnitude of the gradient of the optical flow [15]. Variational calculus is used to minimize an integral and using the mask for Laplacian calculation mean values of the optical flow field components at any point (x,y) read as:

$$\begin{aligned} \bar{u}_{i,j} &= \frac{1}{6}(u_{i-1,j} + u_{i,j+1} + u_{i+1,j} + u_{i,j+1}) + \\ &+ \frac{1}{12}(u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} + u_{i+1,j-1}) \\ \bar{v}_{i,j} &= \frac{1}{6}(v_{i-1,j} + v_{i,j+1} + v_{i+1,j} + v_{i,j+1}) + \\ &+ \frac{1}{12}(v_{i-1,j-1} + v_{i-1,j+1} + v_{i+1,j+1} + v_{i+1,j-1}) \end{aligned} \tag{11}$$

which are used in the pair of equations that relate the optical flow vector from the Laplacian of the image and the spatio-temporal gradients:

$$u = \bar{u} - I_x\frac{I_x\bar{u} + I_y\bar{v} + I_t}{\lambda^2 + I_x^2 + I_y^2} \tag{12}$$

$$v = \bar{v} - I_y\frac{I_xv\bar{u} + I_y\bar{v} + I_t}{\lambda^2 + I_x^2 + I_y^2} \tag{13}$$

where λ is a regularization parameter (a Lagrange multiplier).

Final determination of the optical flow is done from these equations through an iterative process over pairs con changing consecutive images.

4.2 Change-Driven Optical Flow computation

The implementation of the Horn and Schunck algorithm using the strategy based on changes consists on using the equations that treat the data which are involved because of the variation of a given pixel at a certain moment. Differently to the calculation of optical flow in the whole image, it is only calculated for the pixels that have changed most their luminance level.

Initially the algorithm needs the gradient images. Spatial gradients can be calculated from the first frame and temporal gradient (that in classical algorithms is extracted from two consecutive images) is initialized to 0. Since gradients are calculated only from one frame general expressions are particularized to:

$$I_{x_{i,j}} \simeq \frac{1}{2}\left(I_{i,j+1} - I_{i,j} + I_{i+1,j+1} - I_{i+1,j}\right) \quad (14)$$

$$I_{y_{i,j}} \simeq \frac{1}{2}\left(I_{i+1,j} - I_{i,j} + I_{i+1,j+1} - I_{i,j+1}\right) \quad (15)$$

The pixel that has changed most its gray level value is sent by the SCD camera, together with its row and column. After that, the following operations are done:

- Recalculate the spatial gradients for those pixels of the image that are under the influence of the pixel that has changed. According to equations 14 and 15 and looking at the formulae for gradient calculation the involved pixels whose gradients can be modified by the updating of element $(I_{x_{i,j}}, I_{y_{i,j}})$ will be:

$$\begin{aligned} \forall I_{x_{i,j}} &\Rightarrow \left(I_{x_{i,j}}, I_{x_{i,j-1}}, I_{x_{i-1,j}}, I_{x_{i-1,j-1}}\right) \\ \forall I_{y_{i,j}} &\Rightarrow \left(I_{y_{i,j}}, I_{y_{i,j-1}}, I_{y_{i-1,j}}, I_{y_{i-1,j-1}}\right) \end{aligned}$$

- Recalculate the temporal gradient only for pixel (i, j). In this case only the luminance variation for the received pixel is available. This assumes that all other variations are negligible, or that they will be taken into account later when they arrive.
- Do a fixed number of times the following operations:
- Recalculate Laplacian operators $(\bar{u}, \bar{v})$ for all pixel involved by the variation of pixel $(i.j)$. Due to the convolution approximation these elements are taken only as those that surround (x, y).
- Finally use the updated data of gradients and Laplacian to recalculate the new value of optical flow components using equations 12 and 13.

5. Experimental results

Since the development of a physical sensor that implements the change-based approach is still in progress the comparison between the algorithms using a standard frame-based camera and those proposed here has been simulated. It has been assumed a SCD camera with a Winner Take All circuit [16], which selects the row and column of the pixel with the largest change. The grey level of this pixel appears at the sensor output along with its row and column addresses.

5.1 Change-Driven tracking algorithm results

For this experiment, a video that shows a planar disk moving with a non uniform trajectory and velocity has been captured. One of the advantages of the new sensor is the high rate at which changes are delivered, which will allow it to track very fast moving objects. To take this into account the time has been rescaled so that the time difference between frames is taken as $\frac{1}{2000}s$ and the time used to send a change (coordinates and value) is $1\mu s$. This means that between two successive points in time in which the received data are looked the camera could send up to 500 changes.

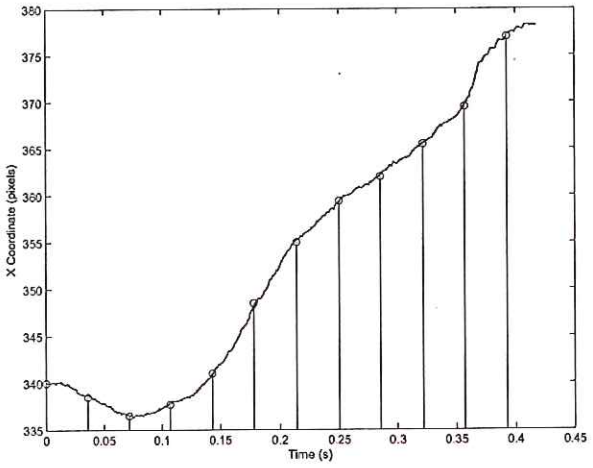

Figure 1: X-coordinate computation for the disk tracking algorithm: vertical lines (classical approach) and continuous line (change-driven approach).

Figure 1 represents a comparison of the X-coordinate spatial tracking obtained by both methods measured in pixels. Points at the top of the vertical lines are those computed with the classical tracking algorithm (full image processing). The continuous irregular line shows all the points computed with the change-driven approach. Using the change-based approach the maximum temporal resolution would be determined by the time required to perform the operations of the algorithm. This is negligible, since it involves only few additions and multiplications for each received change so the simulation of 2000 images per second could even be surpassed and would be limited mainly by the velocity at which the sensor delivers changes. Notice the much higher sampling rate which allows the tracking of objects that move much more rapidly.

5.2 Change-Driven Optical flow results

The well known public Rubik sequence has been used to calculate the optical flow for each frame using 10 iterations and representing the flow vectors with modulus bigger or equal than 0.2; the value of the Lagrange multiplier for the regularization term has been taken as $\lambda = 5$. Results are shown in the figure 2 for the classical algorithm implementation and for the change-driven implementation with different number of pixels. In the change-driven version the optical flow is calculated for every received pixel as long as there is sufficient time until the next integration period at the SCD camera. If not all received pixels can be processed those with a bigger change in their luminance are processed first, since this is the way they will arrive from the real sensor. This can be interpreted as an optical flow calculation at a pixel level instead of at a frame level.

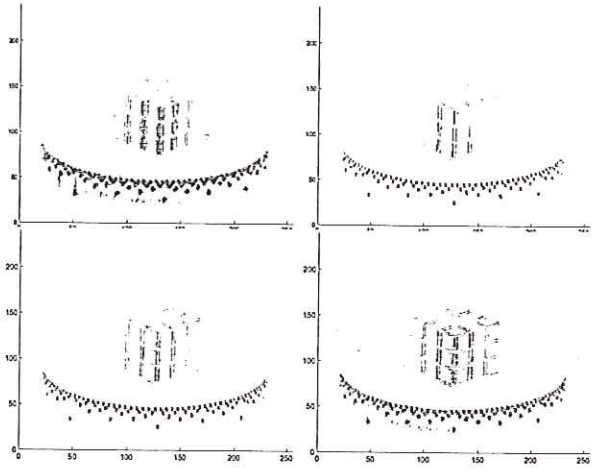

Figure 2: Optical flow computation for the Rubik sequence. From left to right and from top to bottom: original algorithm, change-driven optical flow computation with 1000, 2000 and 4000 pixels, respectively.

Figure 3 shows the mean angular deviation between the original Horn and Schunk algorithm and the change-driven implementation for different numbers of processed pixels. Similarly, figure 4 shows the standard deviation for these mean values. This angular deviation Ψ_E between the real velocity vector components (u_c, v_c) and the calculated velocity vector (u_e, v_e) has been computed through the optical flow error equation:

$$\Psi_E = \arccos\left(\frac{u_c u_e + v_c v_e + 1}{\sqrt{(u_c^2 + v_c^2 + 1)(u_e^2 + v_e^2 + 1)}}\right) \quad (16)$$

The mean error, shown at figure 3, and the standard deviation, shown at figure 4 is not far from most optical flow algorithms. As expected, the error decreases and the accuracy (meaning results similar to the original algorithm) is improved as the number of processed pixels increases.

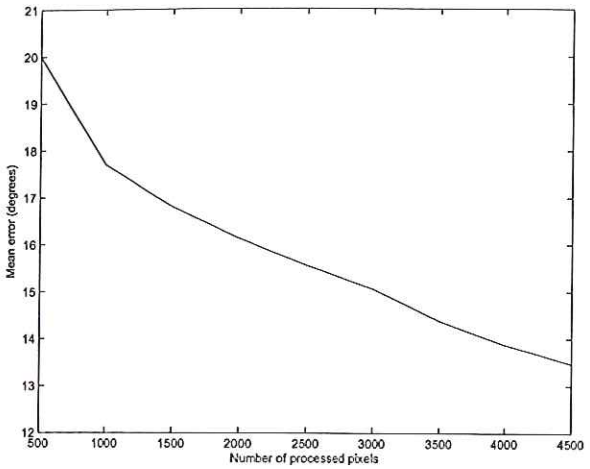

Figure 3: Mean error for the optical flow change-driven algorithm with different number of processed pixels.

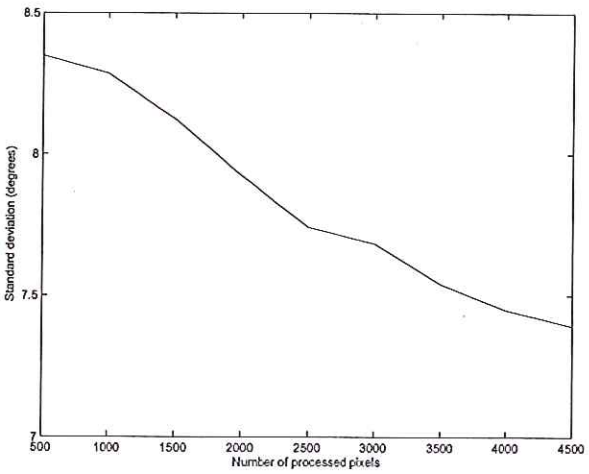

Figure 4: Standard deviation for the optical flow change-driven algorithm with different number of processed pixels.

Experimental measured speed-up of the change-driven method referenced to the classical one is shown in Figure 5. The optical flow is calculated for every received pixel as long as there is sufficient time until the next integration period. As expected, for a low number of processed pixels there is a significant speed-up. If the number of pixels increases then the speed-up decreases. With a number of pixels near to 4000 still there is a speed-up of 1.2 with an error of 13° which can be useful for real-time applications.

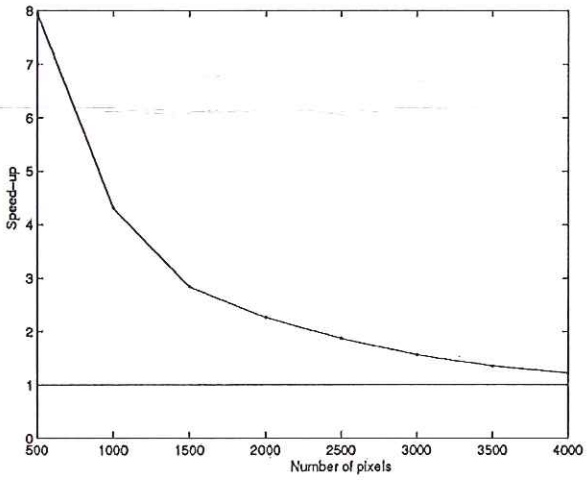

Figure 5: Computation time in function of the number of processed pixels.

6. Conclusion

A change-driven processing instead a full-frame processing for speeding-up motion analysis algorithms has been presented. The change-driven data flow strategy is based on processing the pixels that have changed ordered by its absolute difference value.

This strategy requires an algorithm adaptation (from a pointer based programs to data-flow based models) and extra storage to keep track of the intermediate results of preceding computing stages. Two simple motion analysis algorithms have been chosen to test the change-driven data flow policy: a simple tracking algorithm and the Horn and Schunk optical flow algorithm. Change-driven data flow algorithm implementations show a significant speed-up giving in many cases accurate enough results (similar to the original implementation). Moreover, In the case of the tracking algorithm the change-driven version gives more accurate results than the full-processing version.

7. Acknowledgments

This work has been supported by the project TEC2006-08130/MIC of the Spanish Ministerio de Educación y Ciencia.

References

[1] F. Pardo, J. A. Boluda and E. de Ves, ŞFeature extraction and correlation for time-to-impact segmentation using log-polar imagesŤ, in A. Laganá et al. (Eds.) ICCSA 2004. LNCS, vol. 3046, pp. 887-895, Springer, Heidelberg, 2004.

[2] J. A. Boluda and J. Domingo, "On the advantages of combining differential algorithms and log-polar vision for detection of self-motion from a mobile robot", *Robotics and Autonomous Systems*, vol. 37 (4), pp. 283-296, Dec. 2001.

[3] S. J. Thorpe, A. Delorme, R. Van Rullen and W. Paquier, "Reverse engineering of the visual system using networks of spiking neurons", in *IEEE International Symposium on Circuits and Systems ISCAS*, 2000, Geneva, Switzerland, vol. iv, pp. 405-408.

[4] P, Lichtsteiner, T. Delbruck and J. Kramer, "Improved on/off temporally differentiating address-event imager", in *IEEE International Conference on Electronics, Circuits and Systems ICECS*, Tel Aviv, Israel, vol. 11, pp. 211-214, 2004.

[5] J. A. Boluda and F. Pardo, "Speeding-up differential motion detection algorithms using a change-driven data flow processing strategy", in W. G. Kropatsch, et al. (Eds.) CAIP 2007. LNCS, vol. 4673, pp. 77-84, Springer, Heidelberg, 2007.

[6] F. Pardo, J. A. Boluda, F. Vegara and P. Zuccarello, "On the advantages of asynchronous pixel reading and processing for high-speed motion estimation", in: Bebis, G. et als (eds) ISCV 2008. LNCS, vol. 5358, pp. 205-215, Springer, Heidelberg, 2008.

[7] C. H.Teng, S. H. Lai, Y. S. Chen and W. H. Hsu, "Accurate optical flow computation under non-uniform brightness variations", *Computer Vision and Image Understanding*, vol. 97 (3), pp. 315-346, Mar. 2005.

[8] J. L Martín, A. Zuluoga, C. Cuadrado, J. Lazaro and U. Bidarte, "Hardware implementation of optical flow constraint equation using FPGAs", *Computer Vision and Image Understanding*, vol 98 (3), pp. 462-490, Jun. 2005.

[9] B. McCane, K. Novins, D. Crannitch and B. Galvin, "On benchmarking Optical flow", *Computer Vision and Image Understanding*, vol 84 (1), pp. 126-143, Oct. 2001.

[10] J. J. Xiao, H. Cheng, H. S. Sawhney, C. Rao and M. Isnardi, "Bilateral Filtering-Based Optical Flow Estimation with Occlusion Detection", in *European Conference on Computer Vision ECCV*, Graz, Austria, vol I, pp. 211-224, 2006.

[11] M. Tagliasacchi, "A genetic algorithm for optical flow estimation", *Image and Vision Computing*, vol. 25 (2), pp. 141-147, Feb. 2007.

[12] R. Fransens, C. Strecha and L. Van Gool, "Optical flow based super-resolution: A probabilistic approach", *Computer Vision and Image Understanding*, vol 106 (1), pp. 106-1115, Apr. 2007.

[13] J. Díaz, E. Ros, F. Pelayo, E. M. Ortigosa, and S. Mota, ŞFPGA-based real-time optical-flow systemŤ *IEEE Transactions on Circuits and Systems for Video Technology*, vol 16 (2), pp, 274-279, Feb. 2006.

[14] B. P. K. Horn and B. G. Schunk, "Determining Optical Flow", *Artificial Intelligence*, vol 17, pp. 185-203, Aug. 1981.

[15] B. P. K. Horn, *Robot Vision*, MIT Press, 1986.

[16] G. Indiveri, P. Oswald and J. Kramer, "An adaptive visual tracking sensor with a hysteretic winner-take-all network", in *Int. Symposium on Circuits and Systems ISCAS* 2002, Phoenix, AZ, USA, pp. 324-327, 2002.