

# VLSI Architecture for An Object Change Detector for Visual Sensors

R. Aguilar-Ponce, J. Tessier, A. Baker, C. Emmela,  
J. Das, J. L. TecpanecatI-Xihuitl, A. Kumar, and M. Bayoumi  
Center for Advance Computer Studies  
University of Louisiana at Lafayette  
PO Box 44330, Lafayette LA 70504-4330 USA  
ruth@louisiana.edu

**Abstract—** Object detection is a crucial step in visual surveillance. Traditionally, object detection has been performed purely in software in surveillance systems. The problem of object detection, however, becomes critical in the upcoming wireless visual sensors because of size and power constraints. The need for low-power, small size, hardware implementations is greatly felt. This paper introduces a VLSI architecture for Wronskian Change Detector (WCD). Object detection is done through background subtraction. WCD offers regularity, low complexity and accuracy as well as global illumination changes independency. WCD can be employed in automated visual surveillance on buildings and adjacent parking lots. WCD replaces each pixel by a vector containing luminance value of the pixel and its surrounding area. A linear dependency test is applied to each vector to determine if a change has occurred. WCD is mapped into a 12 – Processing Element array with a fixed window value of  $3 \times 3$ . Design of each processing element is discussed in detail. Based on extensive search, no VLSI implementation of WCD has been reported previously

## I. INTRODUCTION

There is an increasing demand for surveillance system in today's daily life. From the technological-solution perspective, video surveillance has been widely employed for this purpose. Wireless visual sensors promise significant possibilities of performing surveillance at low cost and high speed. The problems of the traditional visual surveillance [1] are further exacerbated by the need to perform low-cost, low-power, and high-speed operations in sensors. These technical challenges include system design and configuration, architecture design, object detection, object identification, tracking and analysis, restrictions on network bandwidth, physical placement of cameras, installation cost, privacy concerns, and robustness to change of weather and lighting conditions. In this work, we focus on the object detection.

Change detection plays a key role in real-time image analysis. Detection on the scene under observation includes moving objects, addition or removal of objects. Therefore, changes due to the change in illumination as well as noise

must be disregarded. One key issue is robustness against illumination changes.

Several approaches have been proposed over the years [2]. The most instinctive technique is frame differencing followed by thresholding. Change is detected if the difference of the corresponding pixels exceeds a preset threshold. The advantage of this technique is its low computational complexity, however it is very susceptible to noise and illumination changes.

Median filter is one of the most popular background subtraction techniques [3]. Median of each pixel of all the frames in the buffer constitutes the background estimation. Background pixels are considered to be those that stay on more than half of the frames on the buffer. However, this technique requires a buffer large enough to store  $L$  frames.

Recursive background techniques do not require a buffer of previous frames. In its place, they recursively update the background model based on each input frame. Any error in the background estimation can remain for a long period due to its recursive nature. The most popular recursive technique is Mixture of Gaussian (MoG) [4]. This method models each background pixel by a mixture of  $K$  Gaussian distributions ( $K$  is a number between 3 and 5). Different Gaussians are assumed to represent different colors. The weight parameter of the mixture represents the time proportions that those colors stay in the scene. The probable background colors are the ones that stay longer and more static. However, the technique is computationally intensive; its parameters require careful tuning and it is very sensitive to sudden changes in global illumination.

Wronskian Change Detector employs the Wronskian of intensity ratios as a measure of change [5]. A large mean or large variance of the intensity ratios increases the Wronskian value. This method can detect object interiors and structural changes. Also, WCD is robust against illumination changes. WCD is a suitable algorithm to be implemented in real-time due to its low complexity. Also, this technique requires only one previous frame; therefore it is appropriate for applications where resources are limited.

A comparison of the discussed methods is presented in Table I. WCD offers a tradeoff between complexity and

storage requirements while achieving medium precision and robustness against global illumination changes. The proposed architecture aims real-time operation while achieving low-power consumption. It achieves real-time operation by performing parallel operation on different pixel values. Low-power consumption is achieved by designing and using low-power components in each processing element.

The organization of the rest of the paper is as follows. Section 2 provides details on the Wronskian Change Detector. Section 3 describes the proposed architecture and components. Section 4 presents the simulation results and conclusions are given in Section 5.

TABLE I. BACKGROUND SUBTRACTION TECHNIQUES

Method	Adaptability	Precision	Complexity	Tuning	Global Illumination Changes	Storage Requirement
FD	High	Low	Low	Simple	Sensitive	1 frame
MF	High	Medium	Medium	Simple	Less sensitive	L frames
MG	Low	High	High	Complex	Sensitive	None
WCD	High	Medium	Medium	Simple	Robust	1 frame

## II. WRONSKIAN CHANGE DETECTOR

Background subtraction technique is widely adopted for surveillance systems due to its low-complexity and ease of use [6-8]. Wronskian Change Detector (WCD) is a non-recursive background subtraction technique that distinguishes changes based on intensity values. Consequently, WCD requires conversion of images into luminance values. Changes are detected based on intensity ratios variance. In order to determine if a change has occurred, a region of support is assigned to each pixel as shown in Figure 1. The size of the region of support can vary from  $3 \times 3$ ,  $5 \times 5$  and  $9 \times 9$  pixels

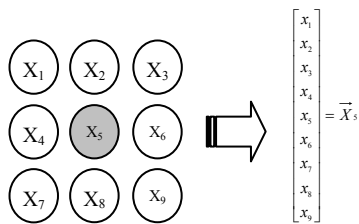


Figure 1. Vector Model for the Wronskian Change Detector

The center pixel is replaced by the vector formed with the pixel and its neighbors. The 8 neighbors are a region of support for the central pixel, in the case of  $3 \times 3$ . When two vectors are linearly independent, then a change can be assumed. A simple and rigorous test for determining the linear dependency of vectors is the Wronskian Determinant. Wronskian Change Detector exploits the fact that the ratio of illuminance values from two light sources helps to quantify the difference between the light sources. The WCD is performed by the equation 1, where  $x_i$  and  $y_i$  are the

components of the vector for the current frame and the previous frame respectively and  $n$  is the dimension of the vector. TH represents the value of the threshold.

$$W\left(\frac{x_i}{y_i}\right) = \frac{1}{n} \left( \sum_{i=1}^n \frac{x_i^2}{y_i^2} - \sum_{i=1}^n \frac{x_i}{y_i} \right) \leq TH \quad (1)$$

$W(x/y)$  detects changes corresponding to dark zones, while its inverse ratio  $W^*(y/x)$  finds if a change has occurred in bright zones. Therefore, computing both values allows robust detection against global illumination changes.

Simulation results show that regions of support larger than  $3 \times 3$  do not provide better results and require more resources. Therefore in our approach a fixed 9-dimension vector has been selected.

## III. PROPOSED ARCHITECTURE

Wireless visual applications impose a strong restriction on power consumption and resources available while requiring real-time operation. Visual applications take video frames coming in NTSC (National Television System Committee) or PAL video standard. The NTSC standard displays 60 fields per second. Each field is composed by even and odd lines. The even and odd fields are displayed sequentially, thus interlacing the full frame. The proposed architecture also accepts frames on PAL standard. PAL standard is the dominant television standard in Europe. The distinction between these standards is that color is handled differently. Since WCD takes region of support containing both fields, processing must be done after storing a complete frame. Processing of incoming frames must be done in 1/30 sec because a camera produces 30 frames per second.

In order to process the incoming frame inside the window frame a processing unit consisting of 12 processing elements working in parallel has been proposed. WCD requires only one previous frame; therefore does not involve a large amount of memory. WCD involves a division resulting in a floating-point precision. However, floating-point implementations are power consuming. Since, power is highly restricted, a fixed-point arithmetic consisting of 8-bit unsigned integer is chosen. Accuracy issues are discussed in the next section.

Our proposed architecture consists of three components: Memory Unit, Controller and Processing Unit. The proposed architecture is depicted in Figure 2. The basic operation of the processor can be divided into three stages: *storing incoming frame*, *processing frame* and *frame output*. The first stage stores the incoming frame in one buffer while the other buffer contains a preceding frame. Upon storing the incoming video frame, both frames are processed in a local neighborhood fashion with the Wronskian determinate. If change is detected for a given pixel, the older pixel in the immediate preceding frame's buffer is overwritten with the new pixel. If no change is detected, the older pixel is

replaced with background. Consequently, when the processing is done, the current frame resides in one frame buffer, and the same image with its background removed (in accordance with the processing results) resides in the other. The processed image can then be transmitted.

This architecture has been designed to ensure flexibility. The system can process a maximum of 10 fps, but if required, fewer frames can be analyzed. This option has been considered because some applications do not need 10 fps; therefore it will be a waste of power to process the extra frames. The image processing clock can be varied, to ensure power efficiency; the processor clock speed can be reduced to as low value as 800 kHz. This benefit comes at the cost of memory, to buffer two entire frames, and the one second delay of the resulting frame. Due to incorporation of enough memory, video output can be done either in NTSC (interlaced) or in VGA (non-interlaced) depending on the video output controller chosen.

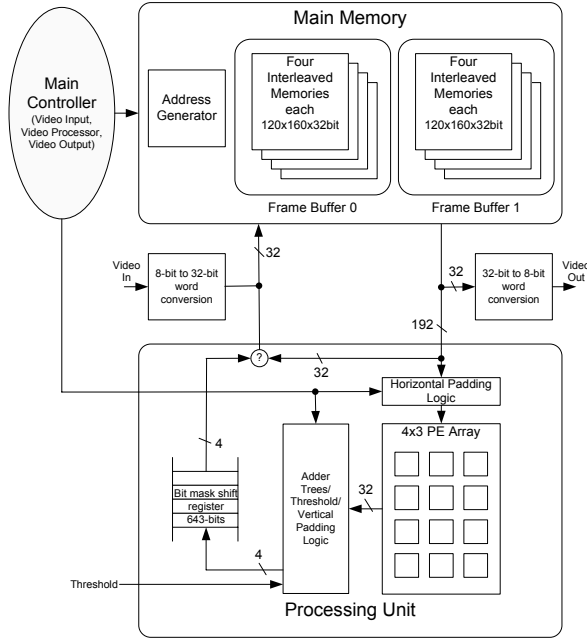


Figure 2. Block Diagram of the Proposed Architecture

### A. Processing Element

The basic operation that must be performed on each pixel value is shown in equation 2.

$$D(x_i, y_i) = \frac{x_i}{y_i} \left( \frac{x_i}{y_i} - 1 \right) \quad (2)$$

Equation 1 can be rewritten as shown in equation 3. Our processing element must perform this calculation to prove what degree of linear independence exists between  $x_i$  and  $y_i$ , which will be used to determine if a change has occurred.

$$W \left( \frac{x_i}{y_i} \right) = \frac{1}{n} \sum_{i=1}^n D(x_i, y_i) \leq TH \quad (3)$$

In designing the PE there are two main concerns, first how to capture the range of the function with only 8-bit unsigned arithmetic. The second concern is guaranteeing precision, considering that [5] suggests threshold values in the range of 0.6 to 0.7 to detect a change. In order to solve these problems, the PE must be designed to capture the range of  $D(x_i, y_i)$  that could indicate a change. Therefore, the equation 3 must be scaled so that an unsigned 8-bit integer threshold can be used and all overflows are saturated. Since the range of the threshold is relatively small, we can achieve near floating-point accuracy. Only the partial range of  $D(x_i, y_i)$  where  $TH_{min} \leq D(x_i, y_i) \leq nTH_{max}$  is significant, where  $TH_{min}$  and  $TH_{max}$  are the minimum and maximum 8-bit threshold to be used. Consequently,  $D(x_i, y_i)$  must comply with the following equation

$$\frac{x_i^2}{y_i^2} - \frac{x_i}{y_i} - nTH_{max} = 0 \quad (4)$$

Solving the equation,

$$\frac{x_i}{y_i} = \frac{1 + \sqrt{1 + 4nTH_{max}}}{2} \quad (5)$$

Using a maximum threshold of 1.7 in a 9-dimensional vector, the 8-bit result of  $x_i/y_i$  could be a 3/5 fix-point number because any result greater than  $4.443 = 1 + \sqrt{1 + 4(9)(1.7)}/2$  would exceed  $nTH_{max}$  and could be saturated. Thus, an equivalent operation for 3/5 fixed point would be

$$D(x_i, y_i) = 2^{10} \left[ \frac{x_i}{y_i} \left( \frac{x_i}{y_i} - 1 \right) \right] \quad (6)$$

This solved the problem of precision, but creates results that are too large to sum  $n$  times. For that reason, the six less significant bit of the product must be discarded after multiplication, and the rest of the bits are employed as the result leading to a final equation

$$D(x_i, y_i) = 2^5 \left[ \frac{x_i}{y_i} \left( \frac{x_i}{y_i} - 1 \right) \right] \quad (7)$$

Since all overflows are saturated at 255, then 254 corresponds to  $TH_{max} = 254/(2^5 n) = 0.8819$  and 0 correspond to  $TH_{min} = 0.0313$ . This yields to a larger range of thresholds,

exclusive use 8-bit arithmetic and 8-bit threshold. The final design of the PE is illustrated on Figure 3.

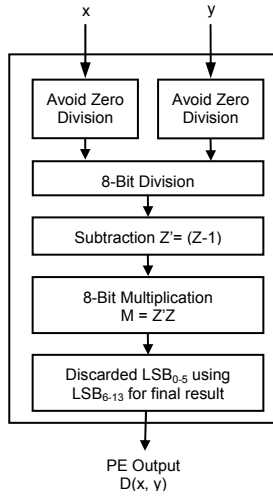


Figure 3. Processing Element Design.

### B. Processing Unit

A slight modification of the basic processing element is illustrated in Figure 4. This modification is labeled as processing element 2. The PE array dimensions are 3 lines by 4 pixels as shown in Figure 5. These dimensions were chosen because 640 is equally divisible by 4.

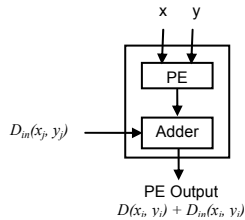


Figure 4. Processing Element 2.

The design uses control signals to pad the image by grounding the bus whenever it is required. The array produces 4 results per iteration with the exception of the first three and the last one results for each iteration. The generation of 4 results per iteration was accomplished by using a fixed window column-wise and a sliding window line-wise. To effectively calculate 3x3 sliding windows, we use two registers to hold partial sums from the last iteration. This reduces the total processing iterations from 307,200 to 76,800.

The adder trees sum PE outputs to produce the final results for four pixels. These results are compared to the threshold, and the 4 change/no change bits are then stored into the bit map shift register. The vertical padding control signal enables/disables summing the last iteration's partial sums. This allows the padding at the beginning and end of each line. The bit map shift register was implemented since calculated change data cannot be stored in the preceding

frame buffer until a particular line of this buffer is fetched for all the corresponding sliding windows. The bit map shift register has two purposes. Firstly, to reorder and group the change bits so that each group of four bits corresponds to each 32-bit word of pixels. Secondly, it buffers one line of results. Since each line must be used at least two times for the sliding window, it is not until the last time a line is used that it can overwrite the frame buffer. The reason for this is, 643-bits of change bits are cheaper to buffer than 5128-bits of pixels especially when the same line must be fetched again.

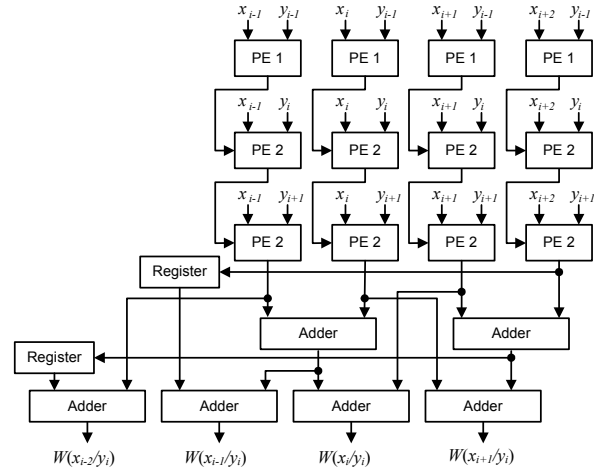


Figure 5. 3 x 4 Processing Element Array.

### C. Controller

The main controller consists of three controllers, *video input controller*, *image processing controller* and *video output controller*. These controllers hand off control to one another. The state diagram depicting the interdependence between these three controllers is shown in Figure 6. The video input controller manages the video input from the video data stream. The architecture supports the analysis of 10 data frames per second. Accordingly, this controller waits for the particular frame from the current second, then stores the data and resets the image processing control unit. We implement our video input controller using a 5-bit counter to determine the number of frames between the reference frame and the current frame to be processed. Once the storing of image is done and if the image processing controller is not reset at that time, the video input controller resets and invoke the image processing controller. The video input controller reaches the active stage upon receiving signal from video output controller.

The image processor controller is implemented to process particular number of pixels per clock cycle of the processing unit. The basic units for image processor controller are two counters, which are used to ensure proper row and column access of the memory unit respectively. These counters also clock the processing units. The image processor controller is invoked by the video input controller, as mentioned above.

Once the processing task is done, this controller will activate the video output controller.

On each cycle of operation, one row is read from the memory in correspondence with the data read from the video stream. Read and write operations are controlled by the address bus controller. Video stream is fed into the system in 8-bit words; therefore a parallel conversion is used to feed data into the processing element array in 32-bit words. A shift register is used for this purpose. The clock rate for the shift register is 4 times faster than the rate for the whole system in order to synchronize them.

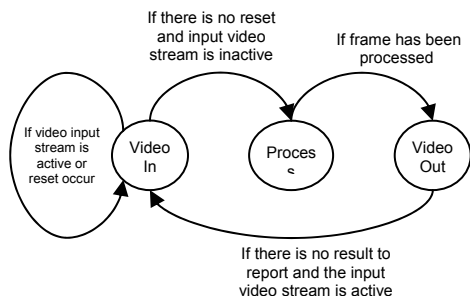


Figure 6. Main Controller.

Video output controller synchronizes the output of the video output. As mentioned above, the architecture uses two memory stages to provide the current video frame data along with the detected change data. Video output controller also selects the output data format, i.e., NTSC or VGA. Once the output operation is done, this controller activates the video input controller, so that video input controller is enabled to take video stream data input

#### D. Memory Unit

The main memory is divided into two stages and each stage contains four interleaved banks, each with the size of at least  $160 \times 120 \times 32$ -bits. An interleaving of at least 3 banks is needed to meet the bandwidth requirement for our architecture that requires 96 bits per iteration. Choosing four banks further lowers the complexity of the address generator. In order to guarantee data fetching from the four windows within one access period, two banks hold every other line of the odd field, and the other two hold every other line of the even field. Such organization also supports the storing of interlaced video data input.

The address generator accesses the memory in two modes: interlaced and non-interlaced. When the video-in is active, i.e., the video input controller is storing the video data, the memory is addressed in interlaced mode. On the other hand, during data transfer to processing units, memory banks are accessed in non-interlaced mode.

## IV. SIMULATION RESULTS

In order to verify the validity of the proposed architecture a preliminary analysis using Simulink 6.1 was done. The

model employed is illustrated in Figure 7. The test images for this simulation are shown in Figure 8, while the results of the simulation on indoor and outdoor scene are shown in Figure 9. The test result was obtained using a threshold value of 0.85 that corresponds to 245 in 3/5 fixed point arithmetic for indoor scene. Results for outdoor scene were obtained with threshold value of 0.2 that corresponds to 58. Test images consist of  $640 \times 480$  pixels.

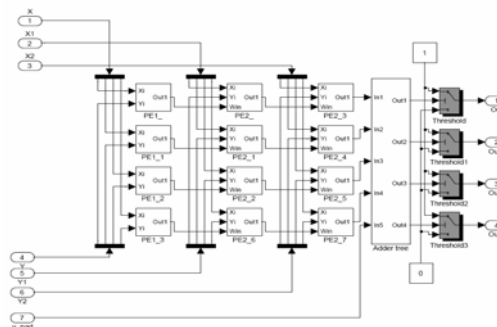


Figure 7. Simulink Model for Wronskian Change Detector.

Implementation of the proposed architecture was done in VHDL using a  $3 \times 3$  window using Mentor Graphic Modelsim Simulator. The main components of the PE are Avoid-Zero Unit, Divider, Adder and Multiplier. Synthesis was done using Synopsis Synthesis Tools targeting  $0.13 \mu\text{m}$  technology. Multiplication is done by Booth algorithm because it represents a good trade-off between speed and power for 8 bit fixed point arithmetic [9]. After reviewing several division algorithms, we conclude that integer restoring using 8 conditional subtractors is simple and fast enough for our application [11].

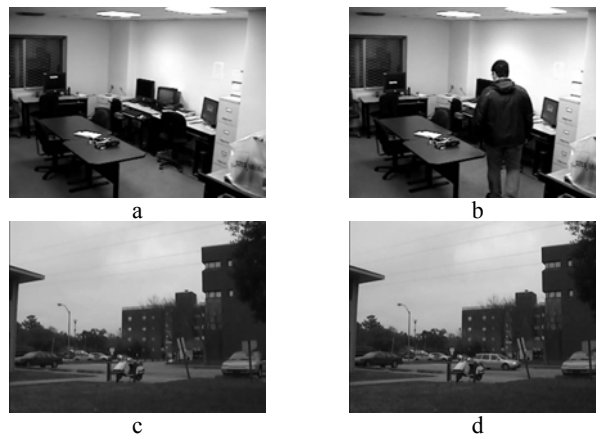


Figure 8. Test Images for Indoor and Outdoor Scene.

The power consumption for a single processing element is 6.69 mW. While the processing element that contains an adder labeled as PE2 consumes 6.89 mW. Table II shows power consumption, delay and area for each component of the processing element. The present architecture operates at 1.2 V with a clock rate 30 MHz. The PE array consisting of

12 elements consume 72.7 mW, while the controller consumes 0.828 mW due its low complexity. Power consumption of the system in the 4 different states is provided in Table III.

TABLE II. PROCESSING ELEMENT POWER CONSUMPTION

Components	Power Consumption (mW)	Delay (ns)	Area (mm <sup>2</sup> )
Avoid Zero Unit	0.136	0.79	0.24
Divider	3.916	12.55	3.69
Booth Multiplier	2.085	3.34	5.49
Adder	0.309	1.12	0.34

Most of the area and power consumption is occupied by the processing unit because of its complexity. The controller unit uses less area due to its simplicity. The architecture consumes 74.08 mW on an average. The total delay is 39 ns and occupied area is 190.9 mm<sup>2</sup>. Most of the power is consumed when the system is processing the frame. If the system is required to process less than 10 fps, then the system will enter in idle state consuming 0.829 mW only.



Figure 9. Results obtained using Simulink, a) Threshold value 0.85 for indoor scene, b) Threshold value 0.2 for outdoor scene.

TABLE III. POWER CONSUMPTION ON THE DIFFERENT STATES

States	Power Consumption (mW)	Delay (ns)
Storing Frame	3.003	2.61
Processing	72.730	19.43
Outputing Frame	3.285	2.60
Idle	0.829	

## V. CONCLUSION

A background subtraction architecture using the Wronskian Change Detector algorithm has been presented for VLSI realization in sensors. The proposed architecture consists of three basic units: controller, memory, and processing unit. Controller is divided into three sub-controllers: video input,

image processing and video output. Video input controller stores the video input stream and indicates when the frame is ready to be processed. The Image Processing Controller directs WCD algorithm execution and points out when the background has been removed. While video output controller sends out video frame with background removed. The processing unit consists of a 12 – PE array that implements WCD algorithm. The memory unit consists of two buffers which store the reference and current frames. The total power consumption for the architecture is 74.08 mW excluding the memory.

## ACKNOWLEDGMENT

The authors acknowledge the support of the U.S. Department of Energy (DoE), EETAPP program DE97ER12220, the Governor's Information Technology Initiative, the DoE award DE-FGO2-04ER46136 and the Louisiana Board of Regents contract DOE/LEQSF (2004-07)-ULL, and the support of National Science Foundation NSF, INF 6-001-006.

## REFERENCES

- [1] A. R. Dick and M.J. Brooks, "Issues in Automated Visual Surveillance", in *Proc. of VIIIth Digital Image Comp. Tech. and App.*, 10-12 Dec 2003, Sydney, pp. 195-204.
- [2] S-C. S. Cheung, C. Kamath, "Robust Techniques for Background Subtraction in Urban Traffic Video". *Proc. Video Comm. and Image Proc.*, SPIE Electronic Imaging, San Jose California January 2004. UCRL-JC-153846-ABS, UCRL-CONF-200706
- [3] R. Cucchiara, M. Piccardi and A. Prati, "Detecting moving objects, ghost and shadows in video stream," *IEEE Trans. On Patt. An. And Machine Inte.* Oct. 2003, pp. 1337-1342.
- [4] N. Friedman and S. Russell, "Image Segmentation in Video Sequences: A Probabilistic Approach," in *Proc. of the 13<sup>th</sup> Annual Conf. on Unce. In Art. Intelligence*, Morgan Kaufmann Publisher Inc. 1997, pp. 175-181.
- [5] E. Durucan and T. Ebrahimi, "Change Detection and Background Extraction by Linear Algebra", *Proc. IEEE*, vol. 89, no. 10, Oct. 2001, pp. 1368-1381.
- [6] R. T. Collins, A.J. Lipton, H. Fujiyoshi, T. Kanade, "Algorithm for Cooperative Multisensor Surveillance", *Proc. of the IEEE*, Vol. 89, No. 10, Oct. 2001, pp. 1456-1477.
- [7] N.M. Oliver, B. Rosario, A.P. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions", *IEEE Trans. on Pattern Anal. and Mach. Inte.*, Vol. 22, No.8, 2000 , pp. 831-843.
- [8] R. Aguilar-Ponce, A. Kumar, J.L. Tecpanecatli-Xihuitl and M. Bayoumi, "An Architecture for Automated Scene Understanding", *IEEE Inter. Workshop on Comp. Arc. for Mach. Perception*, Terrasini - Palermo, July 4-6 2005, in press.
- [9] I.S. Abu-Khater, A. Bellaour and M.I. Elmasry, "Circuit Techniques for CMOS Low-Power High-Performance Multipliers", *IEEE Journal of Solid-State Circuits*, Vol 31, Issue 10, Oct. 1996, pp. 1535-1546.
- [10] C. Kozirakis, D. Judd, J. Gebis, S. Williams, D. Patterson and K. Yelick, "Hardware/Compiler Codevelopment for an Embedded Media Processor", *Proc. of the IEEE*, Vol. 89, Issue 11, Nov. 2001, pp. 1694-1709.
- [11] <http://tima-cmp.imag.fr/~guyot/Cours/Oparithm/english/Divise.htm>