Background Subtraction Algorithm for Moving Object Detection Using Denoising Architecture in FPGA

Anu Susan Philip

M. Tech Student, Dept. of ECE, Mangalam College of Engineering, Kottayam, India

Abstract: ***Currently, in both market and the academic communities have required applications based on image and video processing with several real-time constraints. On the other hand, detection of moving objects is a very important task in mobile robotics and surveillance applications. In order to achieve this, we are using a alternative means for real time motion detection systems. This paper proposes hardware architecture for motion detection based on the background subtraction algorithm, which is implemented on FPGAs (Field Programmable Gate Arrays). For achieving this, the following steps are executed: (a) a background image (in gray-level format) is stored in an external SRAM memory, (b) a low-pass filter is applied to both the stored and current images, (c) a subtraction operation between both images is obtained, and (d) a morphological filter is applied over the resulting image. Afterward, the gravity center of the object is calculated and sent to a PC (via RS-232 interface).***

Keywords: Background subtraction, DTBDM, Similarity module, FPGA, RS-232 interface

1. Introduction

Motion detection is an essential processing component for many video applications such as video surveillance, military reconnaissance, mobile robot navigation, collision avoidance, video compression, path planning, among others. Most of these applications demand a low power consumption, compact and lightweight design, and high speed computation platform for processing image data in real time. In this context, there are three ways for detecting motion in image sequences: (a) background subtraction, (b) temporal difference and (c) optical flow. The most used algorithm is the background subtraction, due to the fact that it is not a computationally expensive algorithm and also presents high performance. The direct execution of hardware algorithms in an FPGA provides speed-up factors typically between 10 and 100 times in comparison with the same algorithm implemented in software, using conventional microprocessors. In this paper, a background subtraction algorithm for motion detection has been implemented in FPGA based board. To accomplish this, a gray level background image is stored in an external SRAM memory (allocated in the FPGA based board). The system performs a post-processing by filtering both the current frame and the background, using spatial convolution before the subtraction. After the subtraction has been performed, the resulting image is segmented using a threshold, and afterward a morphological filtering is applied in order to eliminate the noise of the last stage. Finally, the object's gravity center is calculated and the same is sent to a PC via an RS-232 interface. The RS-232 interface was chosen due to the fact that data traffic is only related to object's position and, therefore, the performance is not affected. Although the system is capable to extract information about object's shape in the motion detection image, in this work measurements about shape are not being performed, given that the system does not have a recognizing stage

2. Theoretical Concepts

Moving object motion detection in image sequences is very important in order to have success in future stages of a computer vision system, such as object tracking, recognition, path planning, among others. The main target of motion detection process is to segment the foreground pixels that belong to the moving objects. To achieve this, there are several approaches for moving detection task, namely (a) the background subtraction, (b) the temporal difference of two successive frames and (c) the optical flow. The background subtraction approach detects the moving regions by subtracting the current frame (pixel by pixel) from a reference image called background that, usually, is found by means of an image selection process, which is executed during a initialization period. The optical flow method is based on the fact that the object motion information is contained in the brightness changes of the image. The three described methods have good performance for motion detection problem, however, optical flow is a very complex algorithm (it is necessary to store more than one image), requiring high memory resources. On the other hand, the background subtraction and the temporal difference are low cost algorithms; however, temporal difference has problems for detecting the object's shape, generally, making difficult a posterior recognition stage. Therefore, in this paper a background subtraction based motion detection system is presented due to his computational low cost, high performance and expressive potential for other applications such as shape detection.

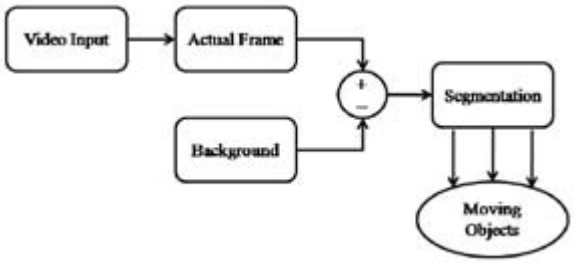


Figure 1: The background subtraction algorithm for moving object detection

Firstly, each image of sequence is subtracted from background. At the same time, the current frame can be used for background actualization (using a control bit). Afterward, the resulting image from the subtraction is segmented in order to produce a binary image that highlights the moving regions on the image that belongs to the moving objects.

Mathematically, the background subtraction algorithm can be defined by (1) , where Td is a predetermined threshold, f(x, y, t) is an image taken at time t and B(x, y) is the reference image (or background). In the dynamic image analysis, all pixels in the motion image d(x, y, t) with value "1" are considered as moving objects .

$$d(x,y,t) = \begin{cases} 1 & \text{if } |f(x,y,t) - B(x,y)| > T_d \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

3. General Architecture

The overall architecture is shown in Fig 2. The general capture system was provided by the board manufacturer, and it includes the CMOS sensor controller and configuration tool, the RAW to RGB conversion and LCD synchronization (using the Multi-port SDRAM controller) and the LCD controller. In this work the color reduction algorithm, the background storage, the motion detection architecture as well as the RS232 communication module have been developed.

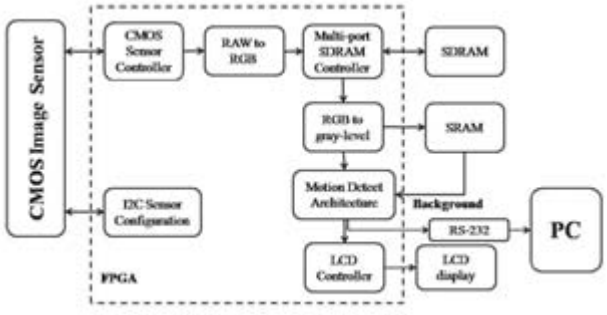


Figure 2: The overall architecture

A. The Color Reduction

The focus of this paper is only the gray-scale image processing and, in this case, the camera provides three color channels of 12 bits. However, the capture control system implemented by the manufacturer allows only 8 bits. There are several methods for determining the appropriate value and in this work, the transformation block calculates the average of the color channels (R, G, B), addressing to subsequent blocks only one pixel. The circuit is synchronized with the clock signal coming from the camera. In this work, the calculations for color reduction are made in a single clock cycle.

B. The Background Storage

The background image is stored in an external SRAM memory

C. The Motion Detection Architecture

In this architecture (see Fig 3) the background subtraction algorithm is implemented after the filtering stage. The filter is applied to both current frame and background images and consists of a low-pass filter (based on spatial convolution process). For this stage the low pass filter has been defined as a mean filter using a 3 × 3 mask. After the filtered current frame is subtracted from the filtered background, the absolute value of resulting pixels is calculated. Afterward, the output image is segmented in order to generate a binary image, where pixels tagged with "1" belong to the moving object and pixels tagged with "0" belong to background. After that, the erosion operation is performed in order to eliminate noise generated by the segmentation operation, and the resulting image of this stage is displayed on an LCD for visualization. Additionally, the object's center is calculated using a gravity center equation and sent to PC, using an RS-232 interface.

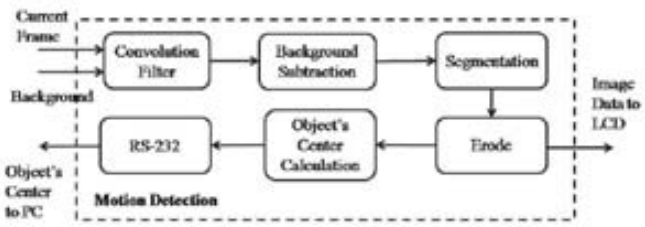


Figure 3: The motion detection architecture

1) The Convolution Filter: The convolution operation is done in one clock cycle, where the rising edge is used to multiply the mask with the neighborhood and the falling edge for yielding the sum of the products.

2) The Background Subtraction and Segmentation Process: Once the filter is performed, the subtraction between the current frame and background one is done, providing one output pixel per clock cycle.

3) The Erosion operation: The erosion operation is based on logic operations between the pixel of a binary image and a structuring element

$$e_i = \bar{K}_i.\bar{f}_i + \bar{K}_i.f_i + K_i.f_i$$

$$Erosion = e_1.e_2.e_3.e_4.e_5.e_6.e_7.e_8.e_9$$

1	1	1
1	1	1
1	1	1

Structuring element used for erosion operation

4) The Gravity Center Calculation: in this work, only one moving object detection is performed and, to accomplish this, the equation (4) is used in order to calculate the object's gravity center C(x, y)

$$C(x,y) = \frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(i,j).B(i,j)}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}B(i,j)} \qquad (4)$$

where B(i, j) is a binary image, i and j are the positions of pixels on the image. The object's gravity center is calculated every frame because of the algorithm has to explore the overall image.

5) The RS-232 Communication: data have to be sent to the PC via RS-232interface.

4. Related Works

A. Modified Works

- Use efficient denoising architecture to remove noise.
- It consists of DTBDM-DT Based noise detector, edge preserving filter.

Images are often corrupted by impulse noise in the procedures of image acquisition and transmission. In this paper, we propose an efficient denoising scheme and its VLSI architecture for the removal of random-valued impulse noise. To achieve the goal of low cost, low-complexity VLSI architecture is proposed. We employ a decision-tree-based impulse noise detector to detect the noisy pixels, and an edge-preserving filter to reconstruct the intensity values of noisy pixels. Furthermore, an adaptive technology is used to enhance the effects of removal of impulse noise.

The Proposed DTBDM

The noise considered in this paper is random-valued impulse noise with uniform distribution. Here, we adopt a 3×3 mask for image denoising. Assume the pixel to be denoised is located at coordinate (i, j) and denoted as pi,j, and its luminance value is named as fi,j, According to the input sequence of image denoising process, we can divide other eight pixel values into two sets: WTopHalf and WBottomHalf.

$$W_{TopHalf} = \{a, b, c, d\}.$$
$$W_{BottomHalf} = \{e, f, g, h\}.$$

DTBDM consists of two components: decision-tree-based impulse detector and edge-preserving image filter. The detector determines whether pi,j is a noisy pixel by using the decision tree and the correlation between pixel pi,j and its neighboring pixels. If the result is positive, edge-preserving image filter based on direction-oriented filter generates the reconstructed value. Otherwise, the value will be kept unchanged

B. Decision-Tree-Based Impulse Detector

In order to determine whether pi,j is a noisy pixel, the correlations between pi,j and its neighboring pixels are considered. Therefore, in our decision-tree-based impulse detector, we design three modules - isolation module, fringe module, and similarity module. Three concatenating decisions of these modules build a decision tree. The decision tree is a binary tree and can determine the status of pi,j by using the different equations in different modules. First, we use isolation module to decide whether the pixel value is in a smooth region. If the result is negative, we conclude that the current pixel belongs to noisy-free. Otherwise, if the result is positive, it means that the current pixel might be a noisy pixel or just situated on an edge. The fringe module is used to confirm the result. If the current pixel is situated on an edge, the result of fringe module will be negative (noisy-free); otherwise, the result will be positive. If isolation module and fringe module can not determine whether current pixel belongs to noisy free, the similarity module is used to decide the result. It compares the similarity between current pixel and its neighboring pixels. If the result is positive, pi,j is a noisy pixel; otherwise, it is noise free.

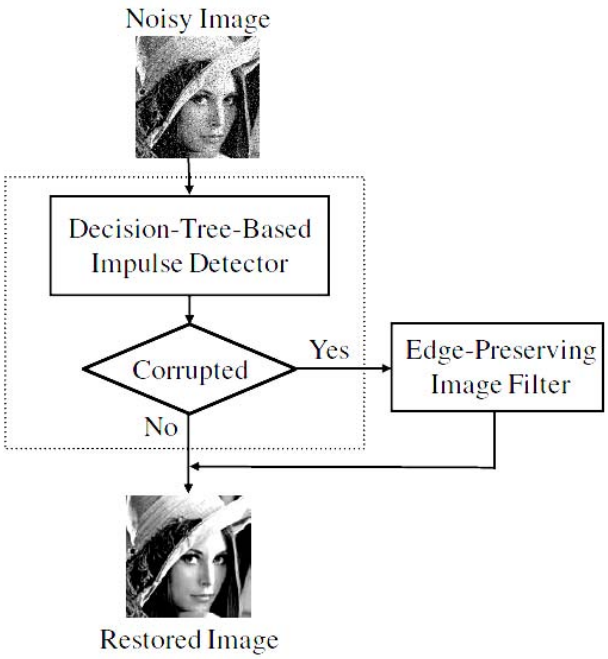


Figure 4: Dataflow of DTBDM

Similarity Module (SM): The luminance values in mask W located in a noisy-free area might be close. The median is always located in the center of the variational series, while the impulse is usually located near one of its ends. Hence, if there are extreme big or small values, that implies the possibility of noisy signals. According to this concept, we sort nine values in ascending order and obtain the 4th, 5th and 6th values which are close to the median in mask W. The 4th, 5th and 6th values are represented as 4thinWi,j, MedianInWi,j and 6thinWi,j.

We define Maxi,j and Mini,j as

$$Max_{i,j} = 6_{th}inW_{i,j} + Th\_SM_a\ .$$
$$Min_{i,j} = 4_{th}inW_{i,j} - Th\_SM_a\ .$$

Maxi,j and Mini,j are used to determine the status of pixel pi,j.

However, in order to make the decision more precisely, we do some modifications as

$$N_{\max} = \begin{cases} Max_{i,j}, & \text{if } (Max_{i,j} \le MedianInW_{i,j} + Th\_SM_b) \\ MedianInW_{i,j} + Th\_SM_b, & \text{otherwise.} \end{cases}$$

$$N_{\min} = \begin{cases} Min_{i,j}, & \text{if } (Min_{i,j} \ge MedianInW_{i,j} - Th\_SM_b) \\ MedianInW_{i,j} - Th\_SM_b, & \text{otherwise.} \end{cases}$$

Finally, if fi,j is not between Nmax and Nmin, we conclude that pi,j is a noise pixel. Edge-preserving image filter will be used to build the reconstructed value. Otherwise, the original value fi,j will be the output. The equation is as:

$$Decision\ IV = \begin{cases} true, & \text{if } (f_{i,j} \ge N_{\max}) \text{ or } (f_{i,j} \le N_{\min}) \\ false, & \text{otherwise.} \end{cases}$$

Edge-Preserving Image Filter: To locate the edge existing in the current W, a simple edge preserving technique which can be realized easily with VLSI circuit is adopted. Here, we consider eight directional differences, from D1 to D8, to reconstruct the noisy pixel value. Directions passing through the suspected pixels are discarded to reduce misdetection. Therefore, we use Maxi,j and Mini,j, defined in similarity module (SM), to determine whether the values of d, e, f, g and h are likely corrupted respectively. If the pixel is likely being corrupted by noise, we don't consider the direction including the suspected pixel. In the second block, if d, e, f, g and h are all suspected to be noisy pixels, and no edge can be processed, so fi, j^ (the estimated value of pi,j) is equal to the weighted average of luminance values of three previously denoised pixels and calculated as

$$(a + b \times 2 + c)/4$$

In other conditions, the edge filter calculates the directional differences of the chosen directions and locates the smallest one (Dmin) among them in the third block. The equations are as follows.

$$D_1 = |d - h| + |a - e|$$
$$D_2 = |a - g| + |b - h|$$
$$D_3 = |b - g| \times 2$$
$$D_4 = |b - f| + |c - g|$$
$$D_5 = |c - d| + |e - f|$$
$$D_6 = |d - e| \times 2$$
$$D_7 = |a - h| \times 2$$
$$D_8 = |c - f| \times 2$$

$$\hat{f}_{i,j} = \begin{cases} (a+d+e+h)/4, & \text{if } D_{\min} = D_1, \\ (a+b+g+h)/4, & \text{if } D_{\min} = D_2, \\ (b+g)/2, & \text{if } D_{\min} = D_3, \\ (b+c+f+g)/4, & \text{if } D_{\min} = D_4, \\ (c+d+e+f)/4, & \text{if } D_{\min} = D_5, \\ (d+e)/2, & \text{if } D_{\min} = D_6, \\ (a+h)/2, & \text{if } D_{\min} = D_7, \\ (c+f)/2, & \text{if } D_{\min} = D_8. \end{cases}$$

$$\bar{f}_{i,j} = \text{Median}(\hat{f}_{i,j}, b, d, e, g).$$

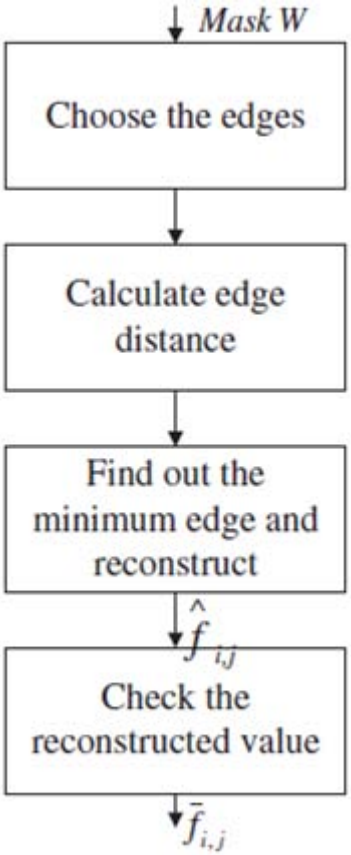


Figure 5: Dataflow of edge-preserving image filter

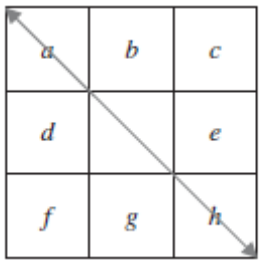


Figure 6: Original image pixel

5. Simulation Results

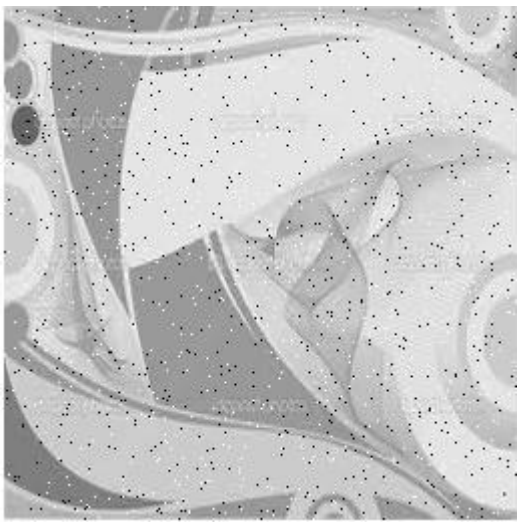
Figure 7: Background image

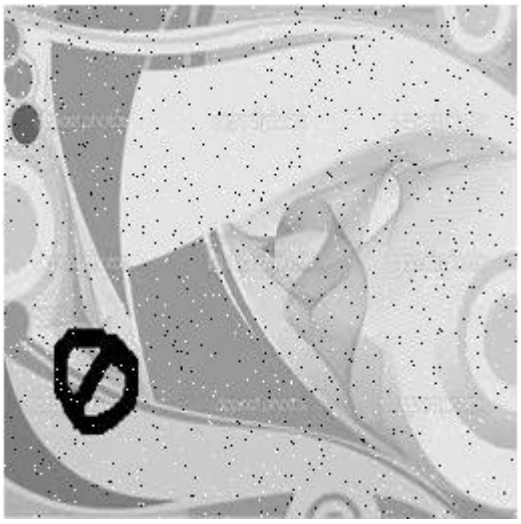
Figure 8: Current image

Figure 9: Output image (After background subtraction algorithm)

Figure 10: To locate the object position(After masking process)

A. Simulation Outputs: Noise Images





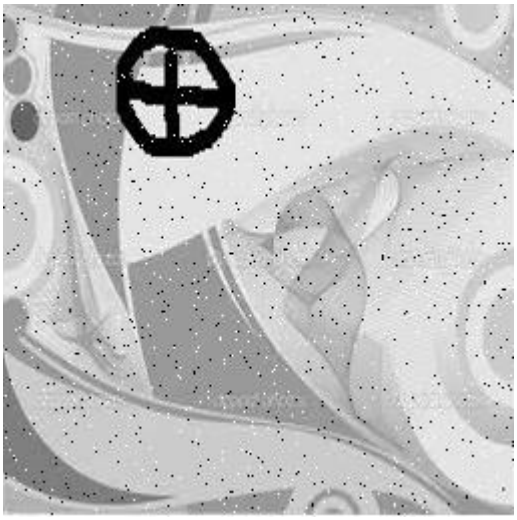


Figure 11: Simulation outputs-noise images

B. Simulation Outputs: Noise Free Images

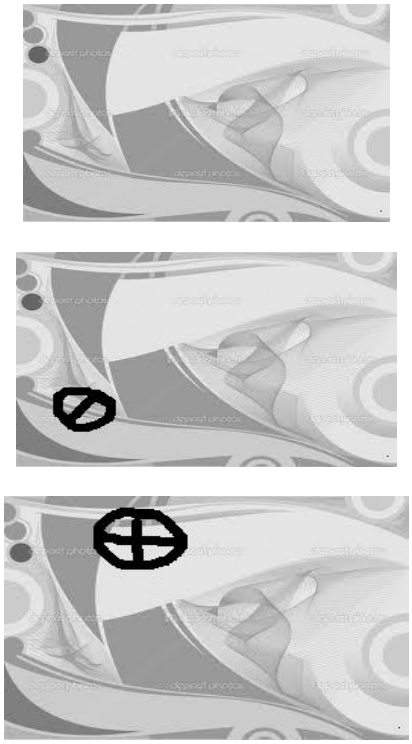


Figure 12: Simulation outputs-noise free images

C. Previous Work Output

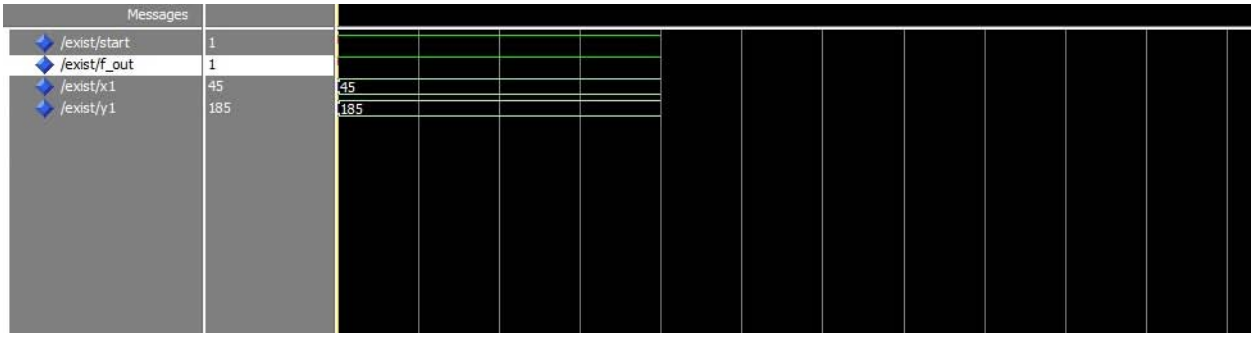


Figure 13: Previous output

D. Modified Work Output

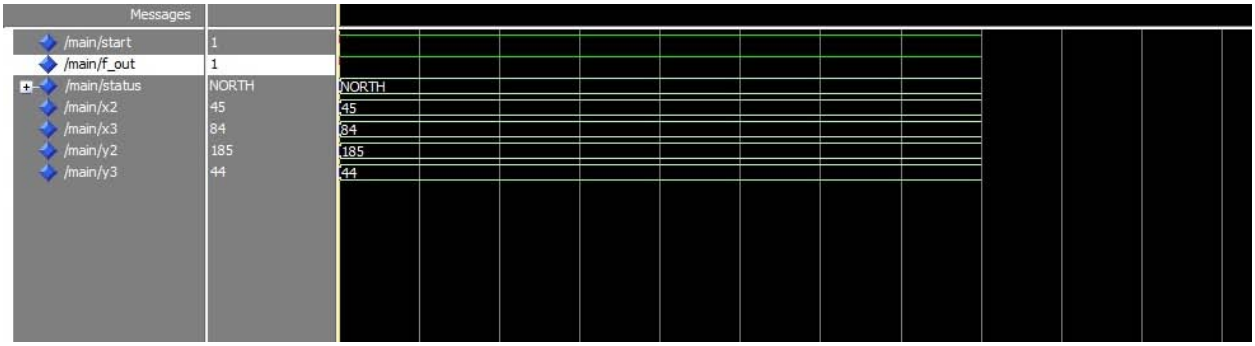


Figure 14: Modified output

6. Applications

Motion detection is mainly used in many video applications-video surveillance, mobile robot navigation, collision avoidance, video compression, path planning.

7. Conclusions

In this work a moving object motion detection system based on background subtraction algorithm was developed. The system works on a real-time pipelined flow. Additionally, the system is capable to detect an object by extracting its shape and calculating the gravity center. The object position is send to a PC or another

platform via RS-232 interface. On the other hand, synthesis results show that area consumption is low, using just 10% of logics elements of FPGA for the overall moving object detection system, allowing the implementation of this system over low-cost FPGAs. The system presents a low-cost architecture that high resources, like memory, are not needed. This fact shows that the implementation of this system on low-cost FPGAs is possible, and it presents good results.

References

[1] E. M. Saad, A. Hamdy and M. M. Abutaleb, FPGA-Based Implementation of a Low Cost and Area Real-Time Motion Detection, 15th International Conference of Mixed Design MIXDES, pp. 249-254, Pozna´n, Poland,2008.

[2] J. L. Mart´ın, A. Zuloaga, C. Cuadrado, J. La´azaro, U. Bidarte, Hardware implementation of optical flow constraint equation using FPGAs, Computer Vision and Image Understanding 98, pp. 462 - 490, 2005

[3] G. G. S. Menezes and A. G. Silva-Filho, Motion Detection of Vehicles Based on FPGA, Proc. IEEE VI Southern Conference on Programmable Logic (SPL), pp. 151-154, Brazil, 2010.

Author Profile

Anu Susan Philip was born in Kollam, Kerala, India. She has received her B.Tech degree in Electronics and Communication Engineering from Kerala University, Kerala, India and pursuing M. Tech in VLSI &Embedded System from Mahatma Gandhi University, Kerala, India. Currently she is the Assistant Professor in Pinnacle school of engineering & Technology, Anchal.