A Real Time Implementation on FPGA of Moving Objects Detection and Classification

K. Sehairi, C. Benbouchama, and F. Chouireb

***Abstract*—** This paper proposes real time implementation on FPGA for moving objects detection and classification using Handel-C language; the results are shown in RGB video format. In the first part of our work, we propose a GUI interface programmed using the Visual C++ that facilitates the implementation for non-initiate users; from this GUI, the user can program/erase the FPGA or change filters or threshold parameters. The second part of this work details the hardware implementation of real-time motion detection algorithm on a FPGA including the capture, processing and display stages using DK IDE. The third part details the algorithm used to classify the moving objects into humans, vehicles and clutter. The targeted circuit was an XC2v1000FPGA embedded on the Agility RC200E board. A PixelStream-based implementation has been successfully performed and completed with a test validation on real-time motion detection and classification.

***Keywords*—**FPGA, Handel-C, Motion detection, Object classification, Real Time, video surveillance.

I. Introduction

Detection of moving objects in video streams is known to be a significant and difficult research problem [1].There are many methods dealing with this problem, temporal difference [2][3][4][5][6][7], background subtraction [8][9][10][11] and optical flow[12][13] . After successfully detecting the moving objects, the problem of identification and classification of these objects follows automatically. There are two main categories of approaches towards moving object classification: Shape-based identification and Motion-based classification [14].

Showing the results of moving object detection in RGB will greatly improve the performance of object recognition, classification, identification and motion analysis and will help significantly the operators to make decisions. However, the computational complexity and the huge information for video involved in object detection and segmentation makes it difficult to achieve real-time performance on a general purpose CPU. There exist many architectural approaches to this challenge: 1) Application Specific Integrated Circuit (ASIC), 2) Parallel Computing, 3) GPUs, 4) DSPs, 5)FPGAs. Evolving high density FPGA architectures such as those with embedded DSP multipliers, memory blocks and high I/O (input/ output) pin count make FPGAs an ideal solution in video processing applications [3].

In our implementation, the detection of objects in motion is based on image segmentation, carried out by calculating the differential image of two consecutive frames [2][3][7][15]. In order to detect the object in motion, the flow of data acquired from camera will be split in two parallel sub-blocks; the first one will be converted from YCbCr to gray-scale and will be used to feed the analysis block. The second sub-block will be converted from YCbCr to RGB and merged with the result of the analysis sub-block. The video has a resolution of 720x576 and the object in motion will be presented in a bounding box around it. For this implementation, we should respect two main constraints: the real time processing and the resources of the targeted FPGA. In addition to that, we were interested in developing a Graphical User Interface in order to send the bit-file that configures our FPGA, erases it or changes a parameter in filter, like the threshold parameter. This helps non initiate users to use the program without the necessity to know the hardware architecture and the IDE.

II. Related Works

Many methods and techniques for motion detection have already been proposed, in [16] they have been classified in three large categories: Background subtraction, temporal difference, optical flow. K.Ratnayake and A.Aishy [3] developed an algorithm for object segmentation and implemented it in Xilinx XC2VP20 using VHDL; they used frame difference algorithm with a spatio-temporal threshold, the design ran at 133Mpixel/s. In another work presented by M.Gorgon, P.Pawlik et al. [8], the authors used the method of Sum of Absolute Differences to detect vehicles for road traffic, the language used was the Handel-C with the PixelStream library of DK Agility, the implementation was done on RC300 board fitted with an FPGA VirtexIIV6000. The results showed that this implementation process 25 standard PAL images in gray scale with resolution of 576x768 in every second, the number of CLBs used is 11%, 5% block RAMs, and 32% of I/O blocks.

K. Sehairi is with the LTSS laboratory, Amar Telidji University, Laghouat, Algeria. An Assistant Professor in Teacher's Superior School of Laghouat ENS-L, Algeria (phone number: +213-661931582; e-mail: k.sehairi@mail.lagh-univ.dz).

C. Benbouchama is an associate professor in Polytechnic Military School, Algiers, Algeria (e-mail: ben_cherrad@yahoo.fr).

F. Chouireb is an associate professor with the Electrical Engineering Department, Amar Telidji University, Laghouat, Algeria (e-mail: chouirebfatima@yahoo.fr).

Another work presented by Wen [4], in which he developed an algorithm to detect and track human motion using the frame difference method with an adaptive threshold, whose parameter was calculated from the difference of smoothed image by Gaussian filter and the original image, the algorithm showed a good result in detecting motion but errors still appeared due to change of brightness or shadow problem. A recent work on frame difference method presented by Wei et al. [7], in which they propose improving this algorithm by using a new method described in four steps: firstly, an absolute differential image is calculated from two consecutive gray frames in image sequences. Secondly, the absolute differential image is filtered by low-pass filter, and translated into binary image. Thirdly, a number of binary images are calculated by some gray images in the image sequences. Finally, motion object region is extracted by arithmetic operations of pixels from binary images mentioned above. This work was not implemented on FPGA, but the result of this algorithm shows that the moving object can be detected precisely. Similarly, Widwayan et al. [17] work on motion detection by combining the frame difference method and Dynamic adaptive template matching in order to increase the accuracy; the image was captured every second from an IP camera and with a resolution of 256x192.The results show that the algorithm provides detection accuracy of 95,5%.

Menezes and Silva-Filho [2] implemented on FPGA the method of background subtraction using a new motion detection architecture ; the goal was the comparison between a software implementation (on an Intel Celeron 2.66Ghz) and a hardware implementation on FPGA SpartanII; the resolution of the video was 126x76, the results showed that the hardware implementation was 7.5 times faster than a software implementation. In [9] L. Ovsenik et al. designed a system of video surveillance in which they used the background subtraction as a first step to detect motion and then track and identify the subject in motion using optical correlator (based on comparing two signals by utilizing the Fourier transforming properties of a lens), the results were applied on detecting an abandoned luggage.

Also, in the classification stage, different descriptions of shape information of motion regions such as representations of point, box, silhouette and blob are available for classifying moving objects [14]. For example, Lipton et al. [6] used the dispersedness and area of image blob as classification metrics to classify all moving object blobs into humans, vehicles and clutter. Another work by Ekinci et al. [18] also used silhouette-based shape representation to distinguish humans from other moving objects and also to recognize the action using skeletonization method. A similar work for pedestrian detection was done by Janta et al. [19] , in which they used the log-gabor filter to extract features and the support vector machines (SVMs) to recognize the pedestrian.

In motion-based identification, we are more interested in detecting the periodic motion of non-rigid human articulated, for example the work presented by Ran et al.[20] in which the authors examine the periodic motion of gait to classify the pedestrian and track it.

For the graphical user interface (GUI), a similar work is done by E.Kobzili et al.[21] in which they used this design for an FPGA implementation of several types of edge detection; the targeted circuit was an FPGA VirtexII embedded on the RC200 board.

III. The Software-Hardware Mixed Design

To make our implementation more flexible, we use the software-hardware platform approach; it simplifies the use of the hardware side and also simplifies the change of data between the soft and the hard, especially for image processing applications that need many parameters to be changed for example the parameters of convolution filter. In our conception, we used the Handel-C language for the hardware part; Handel-C is a behavioral oriented programming language for FPGA HW synthesis and it is adapted to the co-design concept [21].

The software side was developed using the VisualC++ language. After generating the bit file using the Agility DK [22]; we will use the software interface which we have developed to charge this bit file via the parallel port (with a frequency of 50Mhz) on the RC200E board to configure the FPGA. The algorithm parameters will be transferred using this port with the same frequency with 8 bits data length. For the user, these operations are hidden and the graphical user interface allows him to configure or erase the FPGA and to change the algorithm parameters, as an example in our case, we can change the threshold value according to the brightness of the scene.

IV. Outline of the Algorithm

A. PixelStrams library [23]

Before we detail and explain our algorithm and the method used to achieve our goals, we should speak about the tools used for this implementation. We used the RC200E board fitted with an FPGA XC2V1000 [24], this board has multiple video input such as (S-video, Cameran video and composite video) and video output (VGA, Svideo, composite video) and two ZBT SRAM with a 2Mb capacity. The language used is Handel-C [25], and the Integrated Development Environment is DK5 of Agility, and this environment is equipped with different platform development kits (PDK) that contain the Pixel Stream.

Pixel Stream is a library used to develop a system for image and video processing. It comes with many blocks, named filters, which can make a primary video processing like acquisition, streams conversion and filtering. The user has to associate these blocks carefully indicating the type of the stream (Pixel type, coordinates type, synchronization type). Then he can generate the algorithm in Handel-C, and then he has to add or modify the blocks to program his method and he must finally merge the results. It is worth mentioning that these

blocks are parameterizable. That means we can modify the parameters of image processing, for example the size of the acquired image or the parameter of threshold and these blocks are fully optimized and parallelized. Fig. 1 shows the GUI of Pixel Stream.

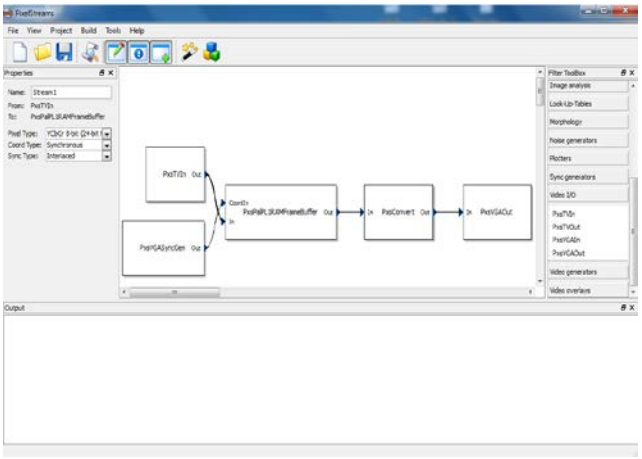

Fig. 1 PixelStream GUI

As mentioned in the introduction, the flow of data will be split in two parallel sub-blocks: an analysis sub-block and a display sub-block. Fig. 2 shows a basic block diagram outlining the algorithmic structure of the object detection process of the system already developed.

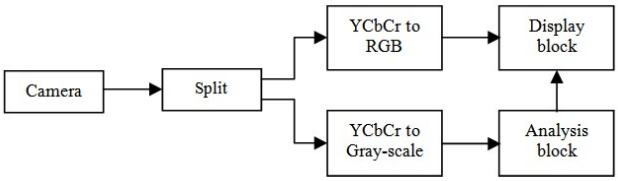

Fig. 2 Object detection process algorithmic basic block diagram

The object detection process can be broken down into the following stages:

B. Image acquisition and splitting

The images acquired from the RC200E board camera, fitted with a Philips SAA7113H video input processor, are in YCbCr color format. These images will be split using the Pixel Stream library of Agility DK, into 2 streams that feed our two sub-blocks. In the display block, we convert the stream from YCbCr to RGB to be able to display it in VGA screen. In the analysis block, we convert the second stream to Gray-scale level to reduce the amount of data analyzed, this will help us to reach the real time constraint and reduce clearly the resource consumed of the FPGA.

There are two ways to do the conversion, either before or after split. If we choose the conversion before split, we have to convert the stream from YCbCr to RGB (such conversion is lossy and should be avoided wherever possible [17]), after that in the analysis block we convert the stream from RGB to gray scale level. But in this case we will increase the number of multiplications to convert the stream from YCbCr to RGB then from RGB to gray-scale level. This conversion can be done by different methods and all these methods use the division operation, which will be avoided especially in FPGA applications. In the average method, we have to divide the sum of three RGB components by three, the lightness method divide the difference between max and min of the three components for each pixel by two and the luminosity method implies taking 30% of the red component plus 60% of the green component and 10% of the third component [ ] (Fig. 3-a).

In the second case, if we choose the conversion after splitting, we have to convert the stream in every sub-block (Fig. 3-b). In the display block we convert the stream from YCbCr to RGB and in the analysis block we convert it from YCbCr to gray-scale. In this conversion, we just take the component Y because it represents the Gray-scale level in the YCbCr format. So we see clearly that we use less resource by avoiding many multiplication operations or division (to convert RGB to gray Scale, case one). Moreover, we are obliged to do approximations in the conversion from YCbCr to RGB, and this conversion is not suitable for the analysis block, that needs all the information captured from the camera. So we choose to let the conversion after split in every branch of sub-blocks. Fig. 3 shows the possible ways to do the conversion; the preferred one is the one presented at the bottom of this figure.

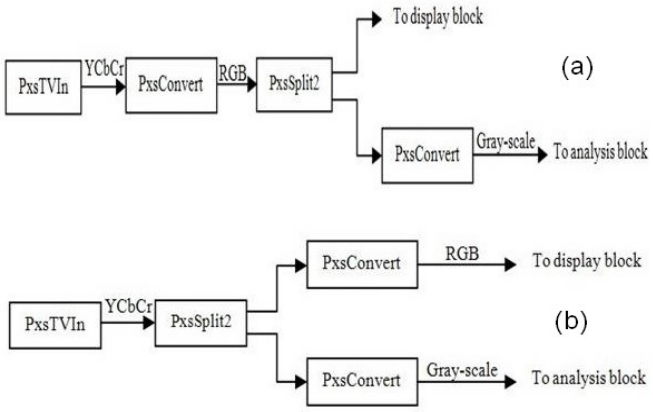

Fig. 3 Possible ways to do the conversion, (a) before splitting, (b) after splitting

C. Analysis block

1. Temporal difference

Temporal difference is a simple method to extract moving objects. It presents an advantage in dynamic environments (example: sun rise). The technique is simple, it consists of saving the first image acquired in frame-buffer $I_{t-1}(x, y)$ and acquiring a second frame $I_t(x, y)$. This second frame will activate the delay cell to diffuse the image stored in the frame buffer according to the coordinate of the pixel. Then the two streams will be synchronized and subtracted (Eq.1). Fig. 4

shows the delay cell and difference diagram.

$$\zeta(x,y)=\left|\frac{dI(x,y)}{dt}\right|=\left|\Delta I_{t,t-1}(x,y)\right|=\left|I_t(x,y)-I_{t-1}(x,y)\right| \quad (1)$$

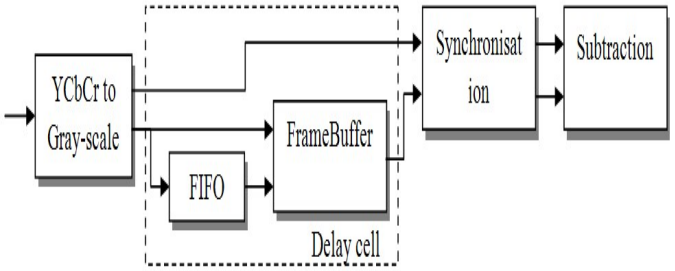

Fig. 4 The delay cell and difference diagram

After that, to get the region of changes, the difference image is thresholded to remove the small changes of luminosity between the two instants (Eq.2).

$$\Psi(x,y)=\begin{cases}0 & \text{if } \zeta(x,y)<Th\\ 1 & \text{otherwise}\end{cases} \quad (2)$$

We should also mention that by a frame buffer we mean the external SRAM in our board RC200E. We have not used the blockRAMs of our FPGA because their capacity is 720Kbits, and this amount lets us deal with an RGB image with a resolution of only 240*128.

2. Block of analysis and statistics

When we get the region of changes from the thresholded difference image, we search in this block the minimum and the maximum along X and Y image axes. Table I shows the code for calculating the Min and Max in Handel-C to give an idea and comparison with the standard C language. In this code, we have used the parallelism instruction (par{}) to calculate these values in one cycle.

Table I. Handel-C program to calculate the Min and Max along the axes X and Y

```
unsigned Minx, Miny, Maxx, Maxy;
Minx=720;Miny=576;Maxx=0;Maxy=0;
If (Value==255) {
par{
Minx=((unsigned 16)xx<= Minx0)?((unsigned 16)xx):Minx;
Miny=((unsigned 16)yy<= Miny0)?((unsigned 16)yy):Miny;
Maxx=((unsigned 16)xx>= Maxx1)?((unsigned 16)xx):Maxx;
Maxy=((unsigned 16)yy>= Maxy1)?((unsigned 16)yy):Maxy;
    }
                }
else {delay;}
Copy the values
Reset all the values
```

It is better to apply a filter before calculating these values to remove the noise; we generally use a median filter or a morphological filter.

To find the center of gravity (COG), we calculate the sum of non-zero pixels coordinates along X and Y axes, and divide this sum by its number. To avoid division we can easily approximate the COG of the object by calculating the COG of the surrounding box, and for that we take the difference between the min and the max in each image axis and shift right one bit to produce the division by two, to which we add the min value (Table II).

Table II. Handel-C program to calculate the COG

```
if (Value ==255) {
   par{N+=1; sumX+= Coord.X;sumY+= Coord.Y;}
                }
else{delay;}
then
xg=sumX/N;
yg=sumY/N;
```

```
The MaxX, minx, MaxY, miny has been already
calculated.
xg=minx+(MaxX-minx>>1)
yg=miny+(MaxY-miny>>1)
```

Once coordinates of the center of gravity are obtained, we copy the values to specific block in the display sub-block where we can plot a rectangle around the detected object. Finally we reset the values of min and max in the two axes.

Fig. 5 shows the result of a temporal difference in our scene obtained using Matlab to visualize the result. We see clearly that the temporal difference gives a non-closed edge of the mobile object. If we search the max and the min along the X and Y axes, we can easily obtain the points to make a bounding box on the object in motion.

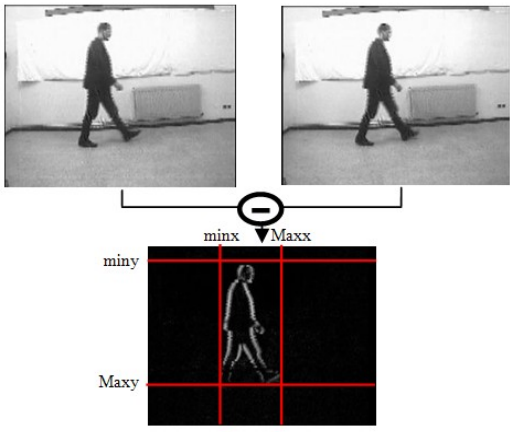

Fig. 5 Temporal difference and the Min and Max in X and Y axis

D. Display block

As stated above, the stream in the display block is converted to RGB. Next we store the image in Frame buffer and wait for the block of analysis to deliver the coordinate of the center of gravity or the four points of the rectangle calculated from the min and max along the two axes X and Y. These points serve us as an input to the block that draws the rectangle.

Using the PxsCursor and PxsConsole we can add a cursor in the center of gravity of the object and add a warning text like "Warning there is motion". For two objects we use two blocks to draw a rectangle around each object. Finally we can copy

stream to VGA output to visualize in a screen. Fig. 6 shows all the diagram of motion detection in RGB.

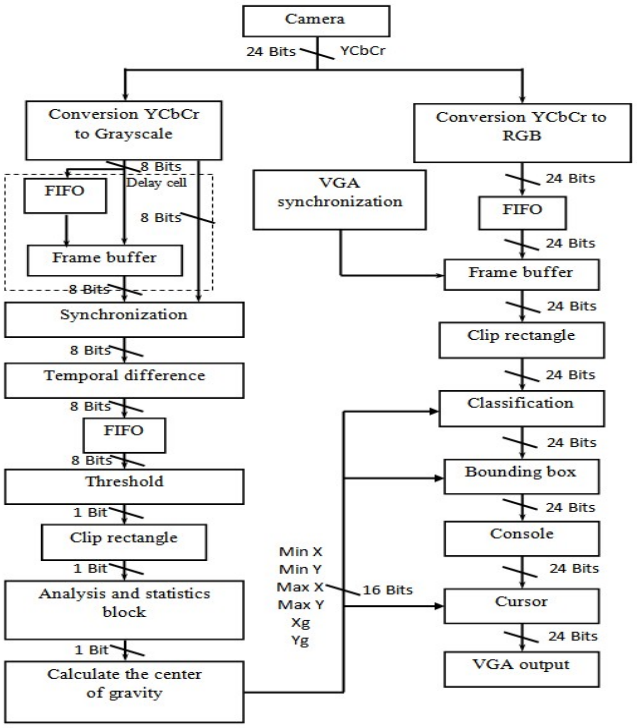

Fig. 6 Diagram of motion detection

V. Object Classification

Moving regions detected in video may correspond to different objects in real-world such as pedestrians, vehicles, clutter, etc. It is very important to recognize the type of a detected object in order to track it reliably and analyze its activities. In our case, the technique we used is based on basic recognition and classification method, the criteria height/width ratio. This method is simple but needs the result of detection to be well segmented. After segmentation, we compute both the width and the height of each detected object, from this ratio we can even specify this moving object whether it is pedestrian or vehicle or other. Figure (11) summarizes the method used for classifying objects.

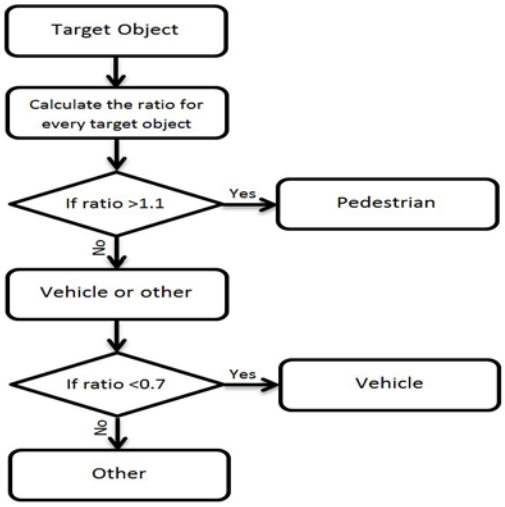

Fig.11.Classification flowchart

For each moving object, we will compute the height/width ratio, if this ratio is superior than 1.1, the moving object is a human being, else we will test the ratio again, if this ratio is less than 0.7 the moving object is a vehicle, else it represents another thing.

The ratios to specify each class are obtained after testing many scenes; figure (12) and (13) show the ratio changes for pedestrian and vehicle respectively for a scene of 50 images, in which, we see that the ratio (height/width) does not fall below 1.1 in case of pedestrian and does not exceed 0.7 for vehicle.

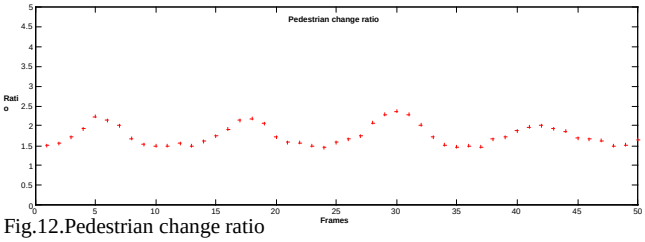

Fig.12.Pedestrian change ratio

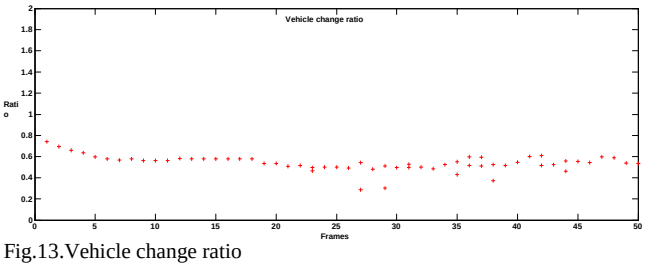

Fig.13.Vehicle change ratio

For implementation of this algorithm, we have already calculated the Min and Max along the X and Y image axis in the analysis sub-block. Using these values, we calculate the Width and the Height for each moving objects. Also to avoid the division we have replaced the condition to detect human beings $(Height/Width) > 1.1$ by the condition $(10.Height > 11Width)$, the same for detecting vehicles the condition becomes $(10.Height > 7Width)$.

VI. Implementation Results

In this section, we demonstrate the effectiveness of the proposed algorithm to work in real time and do the detection in RGB with minimum resources consumed.

Fig. 7 shows the graphical user interface (GUI) that represents the mixed Software/Hardware tool.

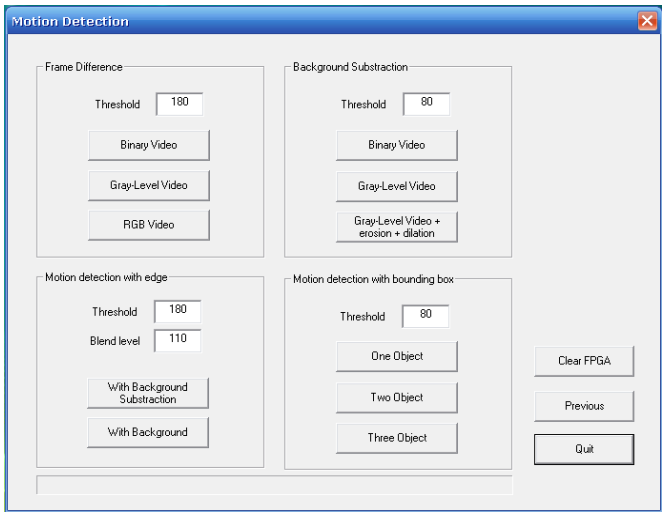

Fig. 7 The graphical user interface

Fig. 8 illustrates the results of our algorithm in scenes, captured from the video result, where a person walks around. The result shows that the person is well detected.

Fig. 8 Motion detection in RGB for one object.

Fig. 9 shows the results for two objects in motion, a warning text displayed when we have motion, and the gravity center coordinates of each object in motion. We can see that the two persons are well detected but when the two objects get closer, the occlusion problem appears and the algorithm considers them as one object.

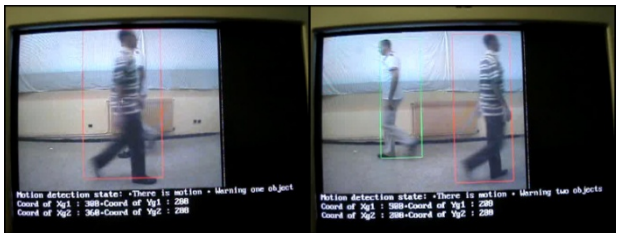

Fig. 9 Motion detection in RGB for two objects.

Figure 10 shows the results of moving object identification divided in three scenes, (a) human identification, (b) car identification and (c) multiple objects identification. The warning text below was done using PxsConsole filter of PixelStream. All the results were filmed using another camera in front of the screen, that's why we can see some blur in our results.

(a)

(b)

(c)

Fig. 10 Classification of detected objects

The resources used for detection and classification of moving objects are demonstrated in table III. This latter shows also the maximal frequency for this implementation. We see clearly that the two major constraints, the technological limits and the real-time aspect (40 ms/image), are respected. This algorithm can treat 70 MPixels per second, it can treat approximately ~170images/s for video size of 720x576 (and it is very far greater than time constraints 25images/s).

Table III. Implementation results

Resources	Total	Resources used
Input/output	324	179 (55%)
LUTs	10240	2,074 (20%)
Slices Flips Flops	10240	2,817 (27%)
CLB Slices	5120	2,844 (55%)
Block RAM	40	9 (22%)
Frequency	/	70.55Mhz (5.88ms/image)

VII. Conclusion

In this paper, we presented a simple method for the motion detection and identification in RGB for real time surveillance application using Handel-C language. Our program was just about 300 lines which proves the ease and rapidity of the Handel-C language, the DK integrated development environment and its libraries on reducing the time of development. In our work we used the most important characteristics in FPGA; parallelism in data and design and computing. Using two parallel sub-blocks helps us to reach the constraints of real time and visualize the detection in RGB; that is more efficient for recognizing the object in motion, unlike other methods that give the result of detection in binary image. For non-initiate users in programming FPGA, the interface is a perfect tool to implement FPGA and it helps the user to change the threshold parameter according to the brightness of the scene.

We have obtained satisfying results, but in the case of two objects the problem of occlusion appears, this is the major problem in all video surveillance applications. In addition, the method based on frame difference is limited; the accuracy of detection changes according to the scene: in indoor areas, the accuracy is better than in outdoor areas due to the complexity of the latter such as the change of luminance, slight motion, and tree branch movement. For future work, we will implement another motion detection algorithm that can operate in complex scenes, and try to recognize these movements and gestures using learning methods.

References

[1] Ying-Li Tian and Arun Hampapur. “Robust Salient Motion Detection with Complex Background for Real-Time Video Surveillance”. In Proceedings of the IEEE Workshop on Motion and Video Computing (WACV-MOTION '05), Vol. 2. IEEE Computer Society, Washington, DC, USA, 30-35.2005.

[2] Menezes, G.G.S.; Silva-Filho, A.G. “Motion detection of vehicles based on FPGA”. Programmable Logic Conference (SPL), 2010 VI Southern , vol., no., pp.151-154, 24-26 March 2010.

[3] Kumara Ratnayake, Aishy Amer. “An FPGA-Based Implementation of Spatio-Temporal Object Segmentation”. In Proceedings of the International Conference on Image Processing, ICIP 2006, October 8-11, Atlanta, Georgia, USA. pages 3265-3268. 2006.

[4] Wen jun-Qin. “An Adaptive Frame Difference Method for Human Tracking”. Advances in information Sciences and Service Sciences(AISS), Volume4, Number1. 2012.

[5] Francesco Ziliani and Andrea Cavallaro. “Image analysis for video surveillance based on spatial regularization of statistical model-based change detection”. Real-Time Imaging 7, 5 (October 2001), 389-399. 2001.
[6] Alan J. Lipton, Hironobu Fujiyoshi, and Raju S. Patil. “Moving Target Classification and Tracking from Real-time Video”. In Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV'98). IEEE Computer Society, Washington, DC, USA,1998.
[7] Wei Shuigen; Chen Zhen; Li Ming; Zhuo Liang, "An Improved Method of Motion Detection Based on Temporal Difference," International Workshop on ,Intelligent Systems and Applications,IEEE, 2009. ISA 2009. vol., no., pp.1,4, 23-24 May 2009.
[8] Marek Gorgon, Piotr Pawlik, Miroslaw Jablonski, and Jaromir Przybylo. “FPGA-based Road Traffic Videodetector”. In Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD '07). IEEE Computer Society, Washington, DC, USA, 412-419. 2007.
[9] L. Ovseník, A. KažimírováKolesárová, J. Turán. “Object Detection in Video Surveillance systems”. Journal of Electronic and Computer Engineering 3 (2010) 137-143. 2010.
[10] Hati, K.K.; Sa, P.K.; Majhi, B., "Intensity Range Based Background Subtraction for Effective Object Detection," Signal Processing Letters, IEEE , vol.20, no.8, pp.759,762, Aug.2013.
[11] Ikhwan H. Muhamad, Fairuz R. M. Rashidi “In-Car Suffocating Prevention Using Image Motion Detection" 2nd proceedings of the 2nd international conference on systems, control, power, robotics (SCOPORO '13). World Scientific and Engineering Academy and Society (WSEAS), Morioka City, Iwate, Japan, 145-150.
[12] Idaku Ishii, Taku Taniguchi, Kenkichi Yamamoto, Takeshi Takaki. ”1000 fps Real-Time Optical Flow Detection System” .Proc. SPIE 7538, Image Processing: Machine Vision Applications III, 75380M (January 28, 2010).
[13] Yassine Benabbas, Nacim Ihaddadene, and Chaabane Djeraba. “Motion pattern extraction and event detection for automatic visual surveillance”. J. Image Video Process. 2011, Article 7 (January 2011), 15 pages. 2011.
[14] Liang Wang, Weiming Hu, Tieniu Tan. “Recent developments in human motion analysis”. Pattern Recognition, Vol. 36, No. 3. (March 2003), pp. 585-601, 2003.
[15] Hazem M. El-bakry, and Nikos Mastorakis “Fast Human Motion Tracking by using High Speed Neural Networks " Proc. of 9th WSEAS International Conference on Signal, Speech And Image Processing (Ssip '09), Budapest, Hungry,September 3-5, 2009, pp. 221-240.
[16] Weiming Hu, Tieniu Tan, Liang Wang, and S. Maybank. “A survey on visual surveillance of object motion and behaviors”. Trans. Sys. Man Cyber Part C 34, 3 (August 2004), 334-352,2004.
[17] Widyawan; Zul, M.I.; Nugroho, L.E., "Adaptive motion detection algorithm using frame differences and dynamic template matching method," Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on , vol., no., pp.236,239, 26-28 Nov. 2012.
[18] Murat Ekinci, Eyüp Gedikli, “Silhouette based human motion detection and analysis for real-time automated video surveillance”. Turk. J. Elec. Eng. & Comp. Sci., 13, (2005), 199-229.
[19] J. Janta, P. Kumsawat, K. Attakitmongkol, and A. Srikaew. 2007. A pedestrian detection system using applied log-Gabor. In *Proceedings of the 7th WSEAS International Conference on Signal, Speech and Image Processing* (SSIP'07), Lang Congyan (Ed.). World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 55-60.
[20] Y. Ran, I. Weiss, Q. Zheng, and L. S. Davis,“Pedestrian detection via periodic motion analysis,” International Journal of Computer Vision, vol. 71, no. 2, pp. 143–160, February 2007.
[21] Kobzili El Houari, El Houari Kobzili, Cherrad Benbouchama, Zohir Irki. “A software-hardware mixed design for the FPGA implementation of the real-time edge detection”. Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on , vol., no.,pp.4091-4095, 10-13 Oct. 2010.
[22] Mentor Graphics Agility (2012). "Agility DK", http://www.mentor.com/products/fpga/handel-c/dk-design-suite/
[23] Mentor Graphics Agility (2012). "PixelStream Manual", http://www.mentor.com/products/fpga/handel-c/pixelstreams/
[24] "Virtex II 1.5v Field-Programmable Gate Arrays", Data sheet, Xilinx Corporation, 2001.
[25] "DK5 Handel-C language reference manual", Agility 2007.