Real-time Implementation of the ViBe Foreground Object Segmentation Algorithm

Tomasz Kryjak
AGH University of Science and Technology,
Krakow, Poland
e-mail: kryjak@agh.edu.pl

Marek Gorgon
AGH University of Science and Technology,
Krakow, Poland
e-mail: mago@agh.edu.pl

***Abstract*—This paper presents a novel real-time hardware implementation of the ViBe (VIsual Background Extractor) background generation algorithm in reconfigurable FPGA device. This novel method combines the advantages of typical recursive and non-recursive approaches and achieves very good foreground object segmentation results. In this work the issue of porting ViBe to a FPGA hardware platform is discussed, two modification to the original approach are proposed and a detailed description of the implemented system is presented. This is the first, known to the authors, FPGA implementation of this algorithm.**

I. Introduction

DETECTION of moving objects (foreground objects) is one of the most important issues in video sequences processing and analysis. It is used in advanced, automated video surveillance systems and traffic monitoring systems, where the robust segmentation of peoples or vehicles is essential to perform reliable tracking or recognition. Intensive work on the mentioned systems can be observed in the image processing researcher community as well as in the industry.

Foreground object detection methods can be divided into three categories: simple successive frame differencing, the so-called background modelling approach followed by background subtraction and optical flow.

In this paper, a hardware implementation of a method belonging to the second of these categories is presented. An extensive review of different approaches to foreground object detection can be found in [3].

The concept of background subtraction involves object detection based on the difference between the current video frame and the background, where the background is understood as an empty scene, i.e. without objects of interest (people, cars). It is worth noting that foreground object detection is not just a simple moving object detection issue. The background may contain moving elements: flowing water, moving leaves and shrubs, which should not be detected. On the other hand, some objects (eg. a pedestrian) may remain still for a while and should be continuously detected. Another source of segmentation errors are objects that start to move (eg. a parked car). The left empty space is than usually misclassified as foreground (a so called ”ghost”). That is why the background representation should be adaptive in order to compensate some normally occurring changes such as lighting or movement of certain objects (i.e. chair in an office), as well as handle difficult cases like ghosts. The process is referred to as background generation or modelling.

In the literature one can find many descriptions of FPGA implementations of background generation methods. An extensive discussion of this issue is presented in [6]. The most important and recent articles are:

- Mixture of Gaussian [4] – HD greyscale video stream processing (1920 × 1080 @ 20 fps),
- Horparsert method [9] — 1024 × 1024 @ 32.8 fps, video stream processing, the high level synthesis language Impulse-C was partially used,
- Codebook [8] — 768 × 576 @ 60 fps video stream processing,
- Clustering [6] – HD colour video stream processing.

An FPGA implementation of background generation algorithms can be used in hardware accelerators (e.g. frame-grabbers with an FPGA device, which perform some image pre-processing and analysis) or smart cameras, where all the image processing, analysis and recognition is performed in the camera and only the results are transmitted to the main processing unit of a surveillance system.

II. The ViBe Algorithm

The foreground object segmentation algorithm ViBe (VIsual Background Extractor) was proposed by O. Barnich and M. Van Droogrnbroeck and described in detail in [1], [2], [12]. It contains several innovative elements (the solution is patented) and allows to obtain very good results, which is confirmed by a high place in the object detection algorithms ranking [5].

The background model in ViBe consists of a set of observed pixel values. This is an important difference compared to the most common methods, where the background model is based on probability distribution function. The authors of ViBe justify this concept, pointing out the difficulties in selecting the appropriate probability distribution and the corresponding update mechanism.

Let $v(x)$ denotes the pixel value in a given colour space at the point x in the image, and v_i the i-th sample from the background model. Then the model for each pixel x is defined as a set of N samples:

$$M(x) = \{v_1, v_2 ..., v_N\} \quad (1)$$

In order to classify the pixel $v(x)$ a sphere $Sr(v(x))$ of radius R centred at the point $v(x)$ is defined. The analyzed pixel is considered as background, if at least $\#_{min}$ samples from the model $M(x)$ lie inside the sphere. The distance is defined as Euclidean, and the procedure requires, in the worst case, N distance calculations and N comparisons.

The authors proposed a method of initializing the background model using a single video frame. This results in fast initialization and re-initialization i.e. in case of a sudden lighting change or surveillance system reboot. In this approach, however, the temporal context (history of the pixel) is not available, therefore, the assumption has been made that the adjacent pixels should have similar values. The initialization procedure involves filling the buffer $M(x)$ with randomly selected samples from the pixel's spatial context (size 3×3).

The disadvantage of this approach is its susceptibility to artefacts in the form of "ghosts" — a collection of pixels classified as belonging to the foreground, but actually not related to any real object. The elimination of such interference provides the discussed below background model update mechanism.

The ViBe algorithm uses a conservative update approach — the background model is modified only in the case of classifying a pixel as part of the background. On one hand, it prevents the penetration of moving objects into the background model, but at the same time it can lead to irreparable segmentation errors (e.g. "empty" space left by a car which drove away is classified as an object).

Contrary to popular background generation algorithms that use a pixel buffer approach (average of the buffer, the median of the buffer) where the update process relays on replacing the oldest sample by a new value (FIFO scheme) in ViBe the temporal context is not considered. The sample, to be updated, is chosen at random. In conjunction with the conservative approach this results in an exponential lifespan of a given sample. To further extend the time interval, which is covered by the background model, the update is performed with a fixed probability (e.g. 1/16).

In order to counteract the negative effects of the assumed conservative approach, a mechanism of updating the adjacent background models was proposed. It can be described as follows. If the current pixel $v(x)$ is regarded as belonging to the background, two update procedures are executed: for the current and the neighbouring background models. First of all, in a random fashion, it is determined whether the update should be executed (the proposed by the authors likelihood equals 1/16). Then, in the first case the sample to be substituted is randomly selected (1 out of N). In the second, the neighbouring model (1 out of 8 assuming a 3×3 context) and the sample (1 out of N) are chosen. The selected samples are then replaced by the value $v(x)$.

It is worth noting that the ViBe method requires very few parameters. The authors of the paper [2] proposed the following values: $N = 20$ (number of samples in the model), $R = 20$ (the radius of the sphere, value for greyscale images), $\#_{min} = 2$ (the minimum number of samples, which must lie within the sphere) and the update probability (1/16) and they were used in the module.

III. Considerations About Implementing ViBe in Hardware

One of the main problems with implementing background generation algorithms in hardware is providing a quick access to the external memory resources, where the background model is stored [6]. In the case of the ViBe algorithm is necessary to ensure the following transfer rate:

$$T = N \times B \times PC \times 2 \tag{2}$$

where: N – model size (number of samples), B – number of bits per one sample (for greyscale images $B = 8$, for RGB $B = 24$, for CIE Lab $B = 23$), PC – pixel clock (for VGA resolution 640×480 PC is 25 MHz). The use of the multiplier two, results from the need to perform write and read operations. Substituting the appropriate values the following rates are obtained: $T \cong 690$ MB/s for greyscale and $T \cong 2070$ MB/s for RGB. In case of HD image processing (i.e. 1920×1080, pixel clock 148.5 MHz) the rates are respectively: $T \cong 4898$MB/s i $T \cong 12293$ MB/s.

Modern FPGA boards are usually equipped with an external DDR3 RAM module. In this study two platforms were analyzed: ML605 (Virtex 6 device) and VC707 (Virtex 7 device), both from Xilinx. The first of these is equipped with a memory with a maximum theoretical transfer rate of 6400 MB/s, and the other 12800 MB/s. Wherein, in the case of dynamic memory, it is impossible to obtain the maximum values, because of the necessity of refreshing and accessing individual banks, and columns.

Analysis of the presented numbers allows to draw the following conclusions. VGA-resolution algorithms can be implemented on both platforms. In the case of an HD video stream only the grey-scale version can be realized on the newer VC707 board. Also, there are a few possibilities to process a HD video stream without increasing the memory bandwidth: processing every n-th frame (lower FPS) or storing for example only one out of four pixels (3 from the 2×2 context are approximated).

The ViBe method can be quite easily implemented in hardware. The distance calculation between the current pixel and the samples in the model is possible to realize in parallel. Other operations, including the pseudo-random number generation are also feasible. Quite complex is only the propagation of the current pixel value to neighbouring models mechanism, which requires the generation of a very wide (more than $N \times B$ bits) context and therefore large number of delay lines - usually implemented in Block RAM memory resources.

IV. The Modifications Proposed to the Algorithm

In the first stage of the research the paper [12], in which the authors propose a series of improvements to the ViBe algorithm was examined in detail. Unfortunately, implementing most of the presented ideas in reconfigurable resources in

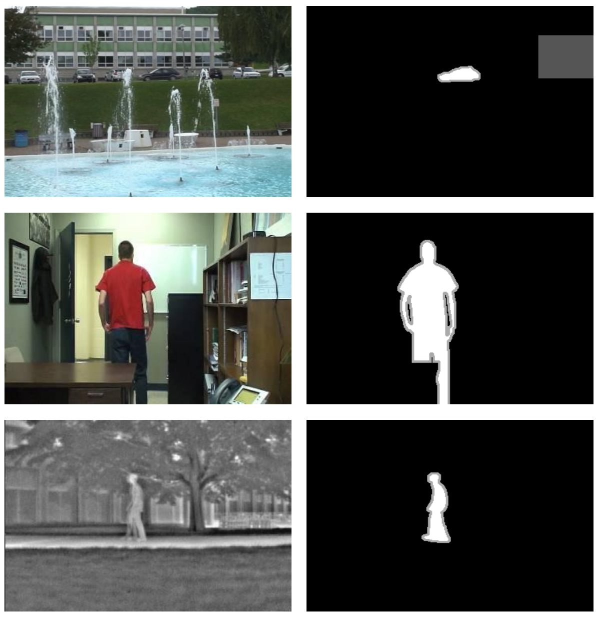

Fig. 1. Sample test images from the `changedetection.net` database. Left coloum - input images, right column - groundtruth. First row — *fountain* sequence (movement in background), second row — *office*, third row — *park* (thermal image)

a pipeline data processing scheme cause huge difficulties or seems to be impossible.

However, in order to enhance the algorithm, two ideas: changing the colour space and a false detection reduction mechanism in areas with background motion (e.g. flowing water) were examined in detail in this paper.

A software model of the algorithm was implemented in C++ using the OpenCV library [7] and examined on the IEEE Workshop on Change Detection (*changedetection.net*) [5] database to evaluate the proposed improvements. The database contains sequences divided into six categories: basic, dynamic background (e.g. flowing river), camera jitter, intermittent object motion, shadows and thermal images. In each of them 4 to 6 videos are included. It can be concluded that the database contains sequences which cover a large part of the situations occurring in surveillance system which are problematic to background generation algorithms. However, the main advantage of the database and a feature that distinguishes it from other collections (e.g. Wallflower [11]), is a large number of manually annotated reference frames with areas marked as: background, shadow, movement, slight blurring and motion (foreground objects). This allows for a reliable assessment of the algorithms performance in different situations. Furthermore, performance results for the most state of the art algorithms are avaliable online (`http://www.changedetection.net/`). Sample images are presented in Figure 1.

The methodology used in the experiments can be described as follows. The object mask computed by the algorithm was compared with the reference mask. Because the ViBe method does not contain a build-in shadow detection procedure, only

TABLE I
PERFORMANCE OF THE VIBE ALGORITHM DEPENDING ON THE USED COLOUR SPACE

Colour space	Distance	PWC [%]	P
Greyscale	L1	3.78	0.67 %
RGB	L1	2.71	0.62 %
RGB	L2	2.28	0.69 %
CIE Lab	eq. (5)	2.18	0.71 %

the foreground and background classification were considered. The following rates were calculated:

- TP (true positive) – pixel belonging to a foreground object classified as a pixel belonging to the foreground,
- TN (true negative) — pixel belonging to the background classified as a background pixel,
- FP (false positive) — pixel belonging to the background classified as a pixel belonging to the foreground,
- FN (false negative) — pixel belonging to a foreground object classified as a background pixel.

Then, based on the calculated parameters two measures were determined: the percentage of wrong classifications:

$$PWC = \frac{FN + FP}{TP + FN + FP + TN} \times 100\% \qquad (3)$$

and precision:

$$P = \frac{TP}{TP + FP} \qquad (4)$$

In the first experiment three colour spaces were examined: greyscale, RGB and CIE Lab. In the first two cases the Manhattan (L1) distance metric was used. Additionally for RGB the Euclidean (L2) metric was calculated. In the case of CIE Lab the fo following formula was used:

$$d_{CIELab} = \alpha \cdot |L_I - L_B| + \beta \cdot (|a_I - a_B| + |b_I - b_B|) \quad (5)$$

where: L_I, a_I, b_I – current pixel in CIE Lab colour space, L_B, a_B, b_B – background model sample in CIE Lab colour space, α, β - weights (in the experiments set to α = 1, β = 1.5). The parameter values for α and β were chosen after evaluation on several test images. The analysis of the CIE Lab colour space was performed due to good segmentation results obtained in a previous work [6]. The experimental results are summarized in Table I.

The presented values are the average rates for the entire dataset. The only modified algorithm parameter was the R threshold. It was set experimentally to obtain best PWC and P ratios.

The results indicate a slight advantage of the CIE Lab over the RGB (L2 metric) colour space. In addition, the hardware implementation of equation (5) is much easier than the Euclidean distance calculation (square and the square root operations require large amounts of FPGA logic resources). Therefore in the final hardware module it was decided to use the CIE Lab colour space, which is a modification to the original proposal from [2].

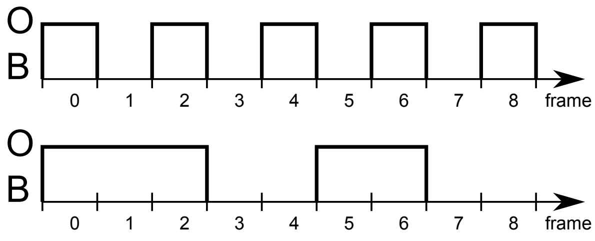

Fig. 2. Two kinds of blinking pixel. O – classification as foreground object, B – classification as background

In the paper [12] an extension to the ViBe method was proposed to detect pixels that are alternatively classified as object and background. They occur most often in cases where small background movement is present (flowing water, fountain, moving grass or leaves). The authors introduced the following mechanism to compensate these interferences. The pixels belonging to the inner boundary of the background that in the previous iteration were classified differently than in the current were detected. In this case the auxiliary variable "blink rate" was incremented by 15, otherwise decremented by 1. If the ratio exceeded the threshold value (set to 30), this pixel was removed from the object mask [1].

This paper proposes an extension to that analysis, which uses two counters: the consecutive classifications of a pixel as background and as an object. This made it possible not only to detect the pixels that change every single iteration (video frame), but also every few ones (Figure 2, bottom graph).

The proposed approach yielded slightly better results. For example, for the "`changedetection.net`" sequence "Canoe" (flowing water) the original approach obtained results: $PWC = 2.19$ and $P = 0.63$, and the proposed $PWC = 1.97$ and $P = 0.68$. It is worth noting that the modification only slightly complicates the algorithm, especially few the hardware implementation.

As post-processing the binary median filter (square window, size 7×7) was selected. It is worth noting thant adding the filter significantly improves the results obtained by the algorithm. An example is presented in Table II.

TABLE II
THE IMPACT OF THE POST PROCESSING MEDIAN FILTERING ON THE ALGORITHMS PERFORMANCE. MEAN RESULTS FOR THE WHOLE DATABASE

Post-processing	PWC [%]	P
none	2.18	0.71 %
median 7×7	1.76	0.88 %

[1]in [12] the mask has been divided into two categories (object and update) and the blinking pixel detection affected only the second. Due to the significant complexity of implementing in pipeline the filling holes operation, which was proposed as post-processing of the update mask, in the presented research this topic was omitted and only one mask was used. For the same reasons the determination of the inner boarder was omitted.

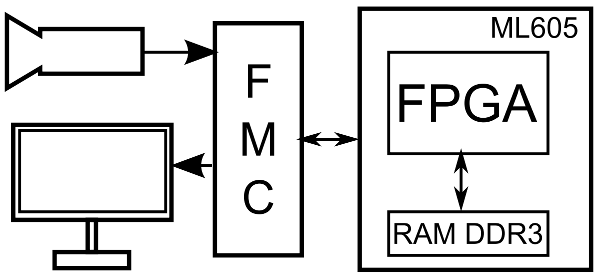

Fig. 3. Scheme of the proposed foreground object detection system

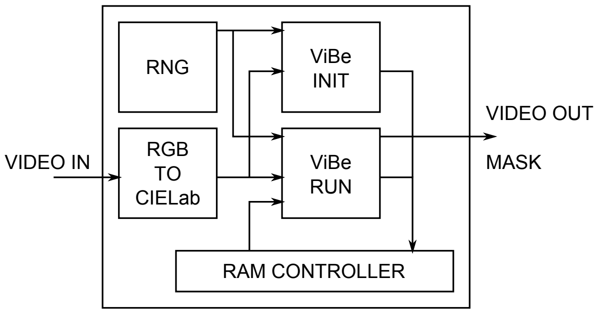

Fig. 4. Block diagram of the modules implemented in the FPGA device

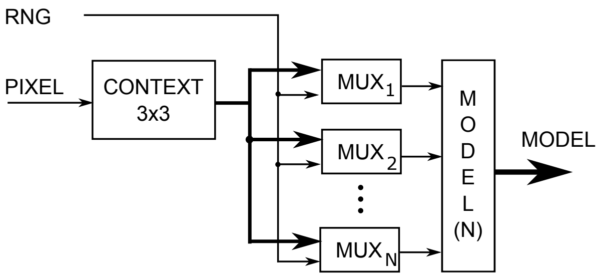

Fig. 5. Block diagram of the *ViBe INIT* module

V. HARDWARE IMPLEMENTATION

Schematically, the proposed system is presented in Figure 3. It consists of an HDMI source (camera or graphic card), HDMI display (LCD screen), Avnet FMC DVI IO module (FPGA Mezzanine Card) with HDMI input and output, and ML605 development board with Virtex 6 FPGA device (XC6VLX240T) from Xilinx. The board is also equipped with an external DDR3 RAM.

All modules were described in VHDL and Verilog hardware description languages. The block diagram of the FPGA design is shown in Figure 4. The *RGB TO CIELab* module is responsible for the colour space conversion [6]. Pseudo-random number generation (*RNG*) was carried out using the concept described in [10]. It is worth noting that the authors made the VHDL code of different RNG versions available, which easy integrates with the project. The used external RAM controller was very similar to the described in [6].

The *ViBe INIT* module is responsible for initializing the background model. The scheme is presented in Figure 5. It consists of a 3×3 context generation module, which uses

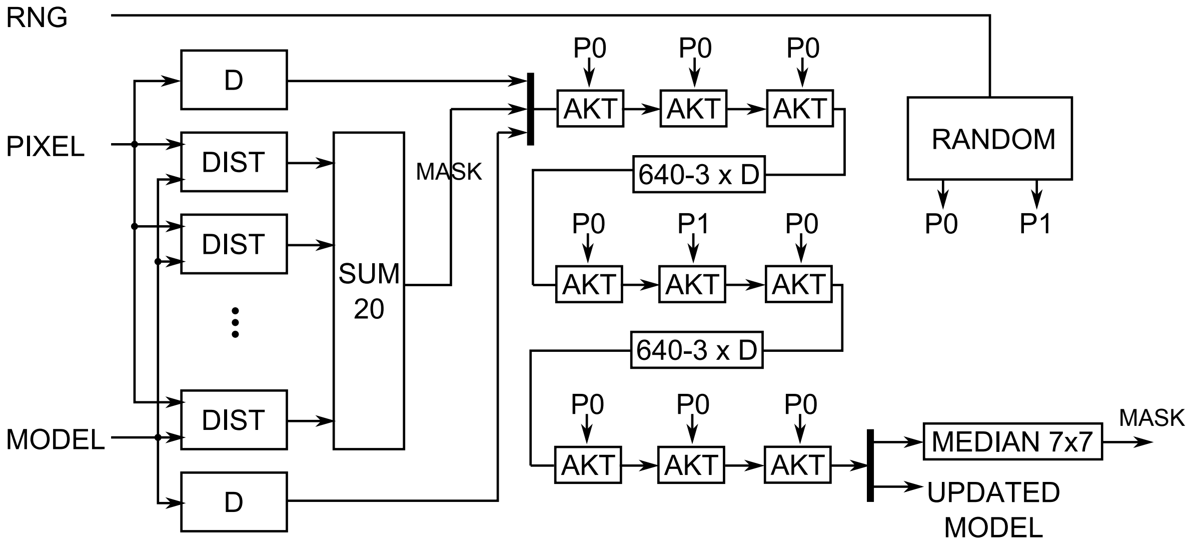

Fig. 6. Block diagram of the *ViBe RUN* module

a delay line approach and N ($N = 20$) multiplexers (*MUX*), which are responsible for the selection of the appropriate sample from the context (1 out of 9 for each *MUX*). The selected value is then stored in the background model. The multiplexers are controlled using a vector obtained from the *RNG* module, thus the model is randomly initialized.

The main module is *ViBe RUN*, which detailed diagram is presented in Figure 6. The inputs are: RNG (pseudo-random number vector), PIXEL (current pixel in the CIE Lab colour space), MODEL (background model read from the external RAM). In the first phase, the distances between the current pixel and the samples from the model are calculated and then compared with the value R (*DIST* - realization of equation (5)). Then it is checked whether the number of distances less than R exceeds the $\#_{min}$ threshold. In the next stage, the 3×3 context consisting of the following signals PIXEL, MODEL and MASK (foreground object mask) is generated. It is worth noting the significant resource usage of this solution - it requires the use of 28 block memory modules (Block RAM). The delay block *D* allows synchronizing the pipeline. The *ACT* module has both a function of a single delay and contains logic that implements the update procedure. The substitution of a background model sample with the current pixel is controlled by the variable P0 (for neighbouring pixels) or P1 (for the central pixel) and depends on the random factor (see Chapter 2) which is schematically illustrated in the form of the *RANDOM* module. The last stage of the process is the median filtering (*MEDIAN* 7×7). The updated model is stored in the DDR RAM and the foreground mask displayed. The blinking pixel detection logic is omitted for clarity reasons.

The presented system was integrated and synthesized for the Virtex 6 FPGA device using the Xilinx ISE Design Suite 14.4. The maximum operating frequency (reported after place & route) was 140 MHz, which is more than enough for processing a VGA colour stream in real-time. FPGA resource

TABLE III
FPGA RESOURCE USAGE

Resource	Used	Available	Percentage
FF	12571	301440	3 %
LUT 6	9278	150720	6 %
DSP 48	13	768	1 %
BRAM_18	172	832	20 %

usage is summarized in Table III. It is worth noting that due to the large context used in the design and buffers required for the DDR RAM controller, the BRAM_18 (Block RAM) utilisation is quite high. The compatibility of the hardware module with the software C++ model was confirmed using the ISim simulation tool.

VI. Conclusion

This paper describes the implementation of the ViBe background generation algorithm in FPGA. Two modifications were proposed: the use of the CIE Lab colour space and improved detection of blinking pixels that have both increased the effectiveness of the method. The results show that, using an appropriate hardware platform, with a fast external RAM, allows implementing in a pipeline manner a quite complex video stream analysis algorithm in real-time. The proposed system enables processing of a colour video stream with a resolution of 640×480 and 60 frames per second. The module can be used in advanced, automated video surveillance systems and other application with require a reliable foreground mask and real-time image processing.

Acknowledgment

This work was supported by the AGH University of Science and Technology grant no 11.11.120.612. The authors would like to thank Mateusz Komorkiewicz for his support in redesigning the DDR RAM controller used in this project.

References

[1] O. Barnich and M. Van Droogenbroeck, "Vibe: A powerful random technique to estimate the background in video sequences," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, 2009, pp. 945–948.

[2] O. Barnichsz and M. Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *Image Processing, IEEE Transactions on*, vol. 20, no. 6, pp. 1709–1724, 2011.

[3] A. S. H. Elhabian S. Y., El-Sayed K. M., "Moving Object Detection in Spatial Domain using Background Removal Techniques - State-of-Art," *Recent Patents on Computer Science*, vol. 1, pp. 32–34, 2008.

[4] M. Genovese and E. Napoli, "FPGA-based architecture for real time segmentation and denoising of HD video," *Journal of Real-Time Image Processing*, pp. 1–13, 2011.

[5] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, June, pp. 1–8.

[6] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "Real-time background generation and foreground object segmentation for high defnition colour video stream in FPGA device," *Journal of Real-Time Image Processing*, pp. 1–17, 2012.

[7] OpenCV, "Strona www: `http://http://opencv.org/` (last acess: May 2013)," 2013.

[8] R. Rodriguez-Gomez, E. Fernandez-Sanchez, J. Diaz, and E. Ros, "Codebook hardware implementation on FPGA for background subtraction," *Journal of Real-Time Image Processing*, pp. 1–15, April 2012.

[9] ——, "FPGA implementation for real-time background subtraction based on horprasert model," *Sensors*, vol. 12, no. 1, pp. 585–611, 2012.

[10] D. Thomas and W. Luk, "FPGA-optimised uniform random number generators using luts and shif registers," in *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, 2010, pp. 77–82.

[11] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, 1999, pp. 255–261.

[12] M. Van Droogenbroeck and O. Paquot, "Background subtraction: Experiments and improvements for vibe," in *IEEE Change Detection Workshop*, 2012, pp. 32–37.