

Real-time background generation and foreground object segmentation for high-definition colour video stream in FPGA device

Tomasz Kryjak · Mateusz Komorkiewicz ·
Marek Gorgon

Received: 31 March 2012 / Accepted: 15 October 2012 / Published online: 7 November 2012
© The Author(s) 2012. This article is published with open access at Springerlink.com

Abstract The processing of a high-definition video stream in real-time is a challenging task for embedded systems. However, modern FPGA devices have both a high operating frequency and sufficient logic resources to be successfully used in these tasks. In this article, an advanced system that is able to generate and maintain a complex background model for a scene as well as segment the foreground for an HD colour video stream ($1,920 \times 1,080$ @ 60 fps) in real-time is presented. The possible application ranges from video surveillance to machine vision systems. That is, in all cases, when information is needed about which objects are new or moving in the scene. Excellent results are obtained by using the CIE Lab colour space, advanced background representation as well as integrating information about lightness, colour and texture in the segmentation step. Finally, the complete system is implemented in a single high-end FPGA device.

1 Introduction

Nowadays megapixels and high-definition video sensors are installed almost everywhere from mobile phones and photo cameras to medical imaging and surveillance systems. The processing and storing of an uncompressed HD

video stream in real-time is quite a big challenge for digital systems.

One of the most fundamental operation in computer vision is the detecting of objects (either moving or still) which do not belong to the background. Knowledge about foreground objects is important to understand the situation that appears in the scene. There are two main approaches: methods based on optical flow (e.g. [5, 11, 25]) and background generation followed by background subtraction. The methods belonging to the second group are the most common to detect motion, assuming that the video stream is recorded by a static camera. The general idea is to find foreground objects by subtracting the current video frame from a reference background image. For almost 20 years of research in this area, a lot of different algorithms were proposed. A comprehensive review on these methods is presented in [7].

When the implementation of a background generation algorithm in FPGA devices is considered, the difference between recursive and non-recursive algorithms has to be stated. The non-recursive methods such as the mean, the median from the previous N frames or W4 algorithm are highly adaptive and are not dependent on the history beyond the N frames. Their main disadvantage is that they demand a lot of memory to store the data (e.g. if the frame buffer $N = 30$ frames, then for the RGB colour images at the resolution of $1,920 \times 1,080$ about 178 MB of memory is needed). In the recursive techniques, the background model is updated only according to the current frame. The main advantage of these methods is that they have little memory complexity and the disadvantage is that such systems are prone to noise generated in the background (they are conserved for a long time). Some recursive algorithms are: the sigma-delta method [26], the single Gauss distribution approach [39], the Multiple of Gaussian (MOG) [37], Clustering [6] and Codebook [21].

T. Kryjak · M. Komorkiewicz (✉) · M. Gorgon
Faculty of Electrical Engineering, Automatics,
Computer Science and Biomedical Engineering,
AGH University of Science and Technology,
al. Mickiewicza 30, 30-059 Kraków, Poland
e-mail: komorkie@agh.edu.pl

T. Kryjak
e-mail: kryjak@agh.edu.pl

M. Gorgon
e-mail: mago@agh.edu.pl

It is important to notice that in the recursive methods, the background model can be represented either by only one set of values e.g. greyscale or three-colour components and some additional data, like variance (the sigma–delta, a single Gauss) or by multiple set of values (MOG, Clustering). The methods belonging to the second group are better suited for scenes with dynamic changes in the lighting conditions (e.g. shadows cast by clouds), multi-modal background and resolve the problem of the background initialisation when there are foreground (moving) objects present in the background training data.

Some work related to the background generation in FPGAs can be found. In the paper [2], an implementation of a method based on the MOG and temporal low-pass filtering taking into consideration the specific character of calculations performed in an FPGA device was described. For each pixel, there were K clusters stored (few possible background representations). A single cluster consisted of a central value c_k (represented by 11 bits: 8 bits integer part, 3 bits fractional part) and weight w_k (6 bits). The following versions of the method were described: a one cluster greyscale ($K = 1$), a one cluster RGB colour space ($K = 1$) and a two cluster greyscale background model (limited by the 36-bit data bus to external memory available on the target platform). The module was implemented on a RC 300 board with a Virtex II 6000 FPGA and 4 banks of ZBT SRAM ($4 \times 2M \times 36$ -bits).

In the paper [19], an implementation of the MOG method in FPGA resources (Virtex 2V1000) was presented. The primary problem with hardware implementation of this algorithm is access to an external RAM—a single Gaussian distribution stored with the selected precision requires 124 bits. Therefore, in the paper, a compression scheme based on the assumption that the adjacent pixels have a similar distribution is proposed. The simulation results show that the method reduces the demand for RAM bandwidth by approximately 60 %.

In the paper [29], foreground object detection using background subtraction was presented. The background was modeled statistically using pixel-by-pixel basis—a single pixel was modeled by the expected colour value, the standard deviation of colour value, the variation of brightness distortion and the variation of chromaticity distortion. Then a histogram of the normalized chromaticity distortion and a histogram of normalized brightness distortion were constructed. From the histograms, the appropriate thresholds were determined automatically according to the desired detection rate. Finally, the type of a given pixel was classified as a background or a foreground object. The algorithm was implemented in an FPGA device and worked with 30 fps with a 360×240 video stream.

In the work [20], the hardware architecture for background modeling was presented. The model used a

histogram for each pixel and as the background, the central value of the largest bin was recognized. This approach was implemented in an Xilinx XC2V1000 FPGA device and allowed to process a greyscale 640×480 video stream at 132 fps. Another approach was introduced in [10]. The background was modeled as an average of consecutive pixels, with a re-scaling scheme. The algorithm was implemented, using the Handle-C language, in a Virtex XC2V6000 device and worked in real-time (25 fps) with a 768×576 greyscale video stream.

In the work [1], an implementation of a sigma–delta background generation algorithm for greyscale images was described. The authors claimed that the system was able to process a 768×576 video stream at 1,198 fps, but they did not consider the external RAM operations and did not present a working solution.

In the paper [11], an implementation of the MOG method in an FPGA was presented. The module processed a high-definition, greyscale video stream ($1,920 \times 1,080$ @ 20 fps). The authors presented results for several different platforms but did not describe a working system (i.e. the external RAM operation, the image acquisition and the display).

In a recent work [35], a hardware implementation of a background generation system in Spartan 3 FPGA device which is using the Horprasert [12] method was presented. Authors added their own shadow detection mechanism which allows to improve the segmentation results. The implementation was performed using the software–hardware approach. Part of the computations was realized in the Microblaze processor. Moreover, two logic description methods were used: high-level Impulse-C language (object detection) and VHDL (secondary modules). In the proposed system, the background was not updated. The initial statistical model was computed by the Microblaze based on 128 learning frames and, when it was created, the system could use it to detect objects. Dilatation and erosion was used in the final processing step (the structural element decomposition method was used). Furthermore, labelling was implemented. The system was able to process images of $1,024 \times 1,024$ resolution at 32.8 fps. The estimated power consumption was 5.76 W.

In another recent paper [34], a hardware implementation on Spartan 3 of the Codebook generation method was presented, which was originally described in the work of [21]. The general idea of the system was very similar to the one described in the article above, only the background generation method was changed. Most important elements were implemented in Impulse-C language. The Codebook algorithm was adapted to allow a fixed point implementation. The system was able to process images of 768×576 resolution at the maximal frame rate of 60 fps. The estimated power consumption was 5.76 W. Finally, comparisons of

segmentation results were presented which points out that the Codebook algorithm has the best accuracy among all other algorithms analysed by authors.

The quality of foreground object detection is very important for further stages of image analysis such as contour retrieving and matching, segment features extraction (area, perimeter, moments etc.), human detection, human action recognition and others. In real life applications, there are many factors which have a negative impact on the quality, namely rapid lighting condition changes (which may affect the background generation) or the foreground object colour similarity to the background. Shadows cast by the foreground objects can also influence the segmentation result.

In this paper, an FPGA-based system which is able to generate the background and extract the foreground object mask for an HD colour video stream in real-time ($1,920 \times 1,080 @ 60$ fps) is presented.

To achieve the high quality of segmentation, several advanced techniques are used: the CIE Lab colour space, a background model with colour and edge information and integration of intensity, colour and texture differences for robust foreground object detection.

The whole system is embedded into a single Virtex 6 FPGA device, with modules implemented for image acquisition, background generation, memory transactions, segmentation and result presentation.

In Sect. 2, the overall concept of the system is described, in Sects. 3–7, particular modules are explained in details. At the end, the results and conclusions are presented.

2 Overview of the system

The proposed system consists of a digital camera, a Xilinx development board ML605 with a Virtex 6 FPGA device (XC6VLX240T) and an HDMI display. In the conception phase, it was decided that all computations should be done inside the FPGA and the foreground objects' mask displayed on the monitor. The idea is presented in Fig. 1.

The system is based on the following elements:

- HDMI video source (e.g. camera or computer with HDMI output),
- Avnet DVI I/O FMC Module with video receiver and transmitter,
- Xilinx ML605 evaluation board with Virtex 6 device,
- HDMI display (e.g. HD TV or LCD monitor).

A functional diagram of the modules implemented in the FPGA device is presented in Fig. 2. Most important are:

- VIDEO IN module responsible for reading the video signal received from the HDMI source by the FMC expansion card,

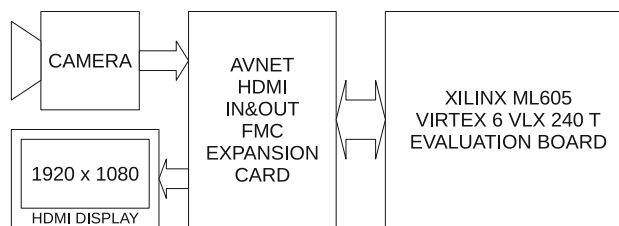


Fig. 1 Overview of the foreground objects detection system

- RGB TO CIE LAB block for changing the colour space from RGB to CIE Lab,
- BACKGROUND GENERATION hardware realisation of an advanced background generation algorithm,
- SEGMENTATION module for foreground objects segmentation,
- VIDEO OUT module responsible for sending the video stream to the HDMI transmitter,
- REGS registers holding parameters needed for the algorithm,
- UART block responsible for transmission of data between registers and PC (run-time adjustments),
- MEM CTRL + FIFO's DDR3 RAM controller with FIFO buffers.

The modules can be divided into two functional groups:

- “algorithmic”(RGB TO CIE LAB, BACKGROUND GENERATION, SEGMENTATION)
- “interface” (VIDEO IN/OUT, MEM CTRL + FIFO's).

All were designed using an HDL language. A detailed description of all implemented modules can be found in Sects. 3, 4, 5.1, 6 and 7.

3 Processing data from camera

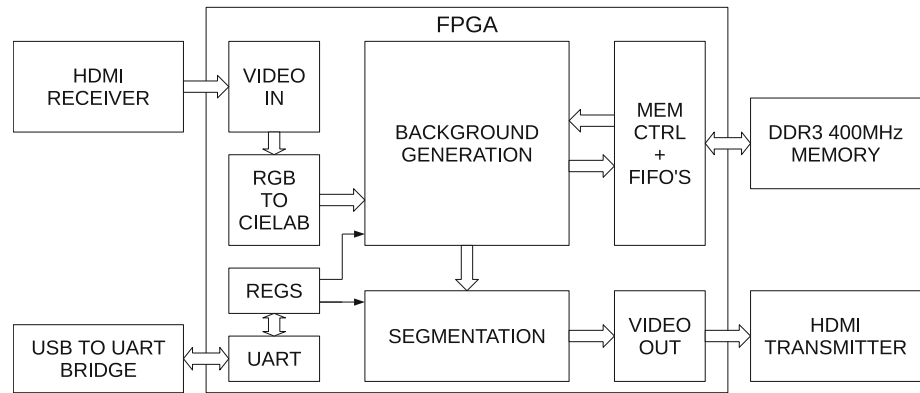
3.1 Receiving data from camera (VIDEO IN)

The module is responsible for receiving a video stream transmitted by the HDMI receiver over the FMC connector to the FPGA. Because the HDMI receiver is transmitting all three colour components in every clock cycle, there is no need for colour restoration (e.g. Bayer filter), however it could be implemented as presented in [22].

3.2 RGB to CIE Lab conversion

When processing colour images, an important issue is the choice of a colour space. In [3], the authors presented research results of different colour spaces. They pointed out that for segmentation combined with shadow removal,

Fig. 2 Scheme of the system implemented in FPGA



the best choices are the CIE Lab or the CIE Luv colour spaces.

In this implementation, it was decided to use the CIE Lab space. In the CIE Lab system, the RGB triplets containing information about intensity of each colour are replaced by L , a , b parameters (L —luminance, a , b —chrominance). Conversion between RGB and CIE Lab is a two stage process [16, 14]. In the first step RGB is transformed to CIE XYZ according to the formula:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.41245 & 0.35758 & 0.18042 \\ 0.21267 & 0.71516 & 0.07217 \\ 0.01933 & 0.11919 & 0.95023 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

The conversion from the CIE XYZ to the CIE Lab colour space is described by the formula:

$$\begin{aligned} L &= 116 * f(Y/Y_n) - 16 \\ a &= 500[f(X/X_n) - f(Y/Y_n)] \\ b &= 200[f(Y/Y_n) - f(Z/Z_n)] \end{aligned} \quad (2)$$

where $X_n = 0.950456$, $Y_n = 1$, $Z_n = 1.088754$ are constants responsible for the white point and $f(t)$ is given by equation:

$$f(t) = \begin{cases} t^{\frac{1}{3}} & \text{for } t > (\frac{6}{29})^3 \\ \frac{1}{3} (\frac{29}{6})^2 t + \frac{4}{29} & \text{otherwise} \end{cases} \quad (3)$$

In order to implement this conversion on an FPGA device, all multiplications were changed to a fixed point and executed on DSP48 blocks of Virtex 6. Because implementing the root functions (Eq. 3) is not possible without using a lot of reconfigurable resources and would introduce a lot of latency, this operations were moved to look-up tables (using the BRAM resource of the FPGA). Since X_n , Y_n , Z_n are constant, four tables were created in which the following values were stored:

$$\begin{aligned} xlut(t) &= 100(t/X_n) \\ ylut(t) &= 100(t/Y_n) \\ zlut(t) &= 100(t/Z_n) \\ llut(t) &= 116(t/Y_n) - 16 \end{aligned} \quad (4)$$

In this way, the problem was transformed into a different form:

$$\begin{aligned} L(X, Y, Z) &= llut(Y) \\ a(X, Y, Z) &= 5 \cdot (xlut(X) - ylut(Y)) \\ b(X, Y, Z) &= 2 \cdot (ylut(Y) - zlut(Z)) \end{aligned} \quad (5)$$

The block diagram of the RGB to CIE Lab conversion module is presented in Fig. 3. The implementation was made using Verilog HDL. The behavioural simulation results are fully compliant to the software model created in C++.

4 Background generation

The starting point for selecting a background generation algorithm were the following assumptions: the algorithm should work with colour images, have a complex background model and the hardware implementation should process a HD video stream in real-time. The use of colour should improve the quality of image analysis and the complex model approach should allow the background modelling to be adaptive both to rapid and slow light condition changes and work properly in case of multimodal background.

An analysis of previous works, as well as preliminary research and tests of the module designed for the Spartan 6 platform [22], have shown that the most crucial constraint that has to be dealt with when implementing a background generation algorithm, is the efficient external memory access. Therefore, the starting point for the algorithm selection was an extensive analysis of memory requirements for several pre-selected methods: single Gaussian (SG); as the method uses a simple background model, the results are presented to illustrate the difference between simple and complex methods [39], multiple of Gaussian [37] and Clustering [6]. The numbers are presented in Table 1. During the analysis the following assumption were made: the video stream resolution was $1,920 \times 1,080$ pixels, a grey-scale pixel or colour component was represented as a fixed point number (8 bits for the integer part and 3 bits fractional

Fig. 3 RGB to CIE Lab conversion module

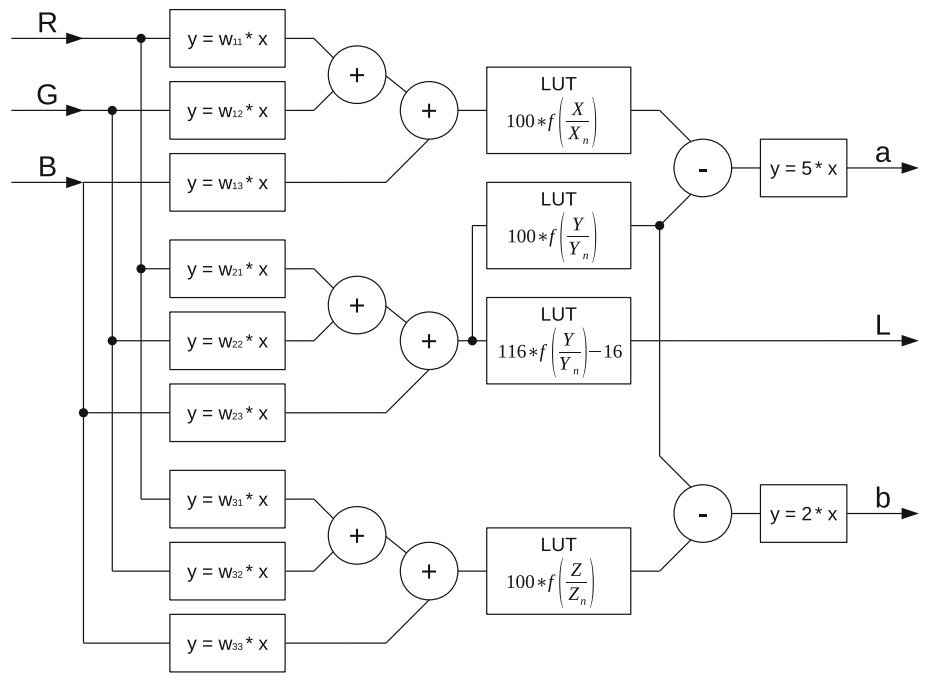


Table 1 Analysis of the required RAM resources for selected background generation methods

Background model	Colour model	Pixel model (b)	Weight (b)	Additional (b)	Single cluster (b)	Location model (b)	The whole model (MB)
SG	Greyscale	11	–	–	11	11	2.72
	Colour	33	–	–	33	33	8.16
SG (with STD)	Greyscale	11	–	11	22	22	5.44
	Colour	33	–	33	66	66	16.31
Clustering (K = 3)	Greyscale	11	6	–	17	51	12.60
	Colour	33	6	–	39	117	28.92
MOG (K = 3)	Greyscale	11	6	11	28	84	20.76
	Colour	33	6	33	72	216	53.39

part), the weight (in MOG and Clustering methods) was represented as 6-bit integer number and the number of clusters (in MOG and Clustering methods) was $K = 3$. The additional bits contain a standard deviation in SG and MOG methods. Then for each method the following values were calculated: size of a pixel model (number of bits required for a single pixel model), size of a single cluster (number of bits required to model a single cluster = pixel model + weight + additional), size of the location model (number of bits required to model one pixel of the background; for SG this value equals “Single cluster”, for Clustering and MOG it equals “Single cluster” multiplied by the number of clusters), size of the whole model (size of the location model multiplied by the image resolution).

Each background model has two important parameters. The first one is the size of the whole model and for the tested methods it ranges from 2.7 to 53.4 MB. For Clustering and

MOG, the model would not fit into the local RAM resources available in the FPGA device (the largest Xilinx Virtex 7 device provides 8.5 MB of BlockRAM), therefore, the use of external RAM is required. The second, and more important parameter, is the size of the location model. In a real-time implementation, the model should be read and written with the pixel clock frequency. For HD resolution video stream (pixel clock 148.5 MHz), it results in a rather high requirement for memory throughput (i.e. for model size 216 bit, 4,000 MB/s). The maximal, theoretical width of the background model for a single location on the target implementation platform (ML 605 board with Virtex 6 FPGA device) was 205 bits (detailed calculations are presented in Sect. 6). Therefore, the only possible choice was the Clustering algorithm [6], as it offers a complex background model, colour processing and the location model size is 117 bits (for the initially assumed precision).

In the presented implementation, some changes to the algorithm described in [6] were introduced. The first one was picking the CIE Lab colour space as better suited for shadow detection and removal according to [3]. The second one was adding information about edge magnitude to the background model. In papers [4, 18], it was pointed out that edges improve segmentation results, particularly in the case of sudden, local illumination changes. The Sobel operator was used to extract edges and the edge magnitude was defined as:

$$\Delta M = |S_x| + |S_y| \quad (6)$$

where S_x and S_y are the vertical and horizontal Sobel gradient, respectively. The value of ΔM was used as the fourth feature together with the L, a, b components. The performed software tests showed that the use of edges reduces the penetration of moving objects into the background model. An example is presented in Fig. 4.

4.1 Evaluation of the background model with edges

In order to evaluate the impact of adding information about edge magnitude into the background model on the background generation algorithm and the segmentation results, tests were performed on the Wallflower [38] dataset. It consists of seven different video sequences, which test several aspects of background generation algorithms:

- “Bootstrap” (B) background model initialization with moving object present in the scene,
- “Camouflage” (C) the object is very similar to the background,
- “Foreground Aperture” (FA) a stationary object starts to move after some time,
- “Light Switch” (LS) sudden illumination changes,
- “Time of Day” (TD) gradual illumination changes,
- “Waving Trees” (WT) small movement in the background,
- “Moved object” (MO) change in the background (moved chair).

For each sequence, a single random frame was manually segmented. The obtained ground truth is then compared

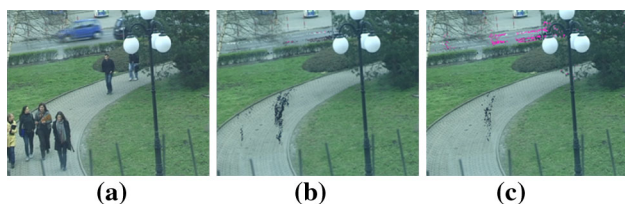


Fig. 4 Example of improving the background generation algorithm by adding edge information to the model. **a** Scene, **b** current background, **c** current background obtained with edge information

with the segmentation result returned by the algorithm under test. The “Moved Object” sequence was not analysed in the quantitative results, because the ground truth does not have any foreground pixels.

The performed evaluation was based on true and false positives and negatives:

- “True Positive” (TP) a pixel belonging to an object is detected as an object,
- “True Negative” (TN) a pixel belonging to background is detected as background,
- “False Positive” (FP) a pixel belonging to background is detected as an object,
- “False Negative” (FN) a pixel belonging to an object is detected as background.

From this parameters, the following measures can be obtained: *Recall* (R), *Precision* (P), *F1* and *Similarity* [13, 34].

$$R = \frac{TP}{TP + FN} \quad (7)$$

$$P = \frac{TP}{TP + FP} \quad (8)$$

$$F1 = 2 \frac{PR}{P + R} \quad (9)$$

$$\text{Similarity} = \frac{TP}{TP + FP + FN} \quad (10)$$

Recall is the true positive rate and it measures the capability of an algorithm to detect true positives. *Precision* measures the capability of avoiding false positives. The *F1* and *Similarity* measures are introduced to evaluate the overall quality of the segmentation. The obtained test results are presented in Table 2.

The analysis of the obtained results shows that in every case using information about edges improves the segmentation results. It can be particularly well observed in the “Time of Day” sequence. This is why adding edge information to the background models is well justified, especially since gradients can also be used in the foreground object segmentation stage (see Sect. 5). Also the visual inspection of the generated background (Fig. 4) confirms

Table 2 Evaluation of the background model with edges

Test	<i>F1</i>		<i>Similarity</i>	
	EDGES	NO EDGES	EDGES	NO EDGES
B	0.71	0.662	0.550	0.495
C	0.946	0.940	0.898	0.886
FA	0.668	0.667	0.502	0.501
LS	0.645	0.634	0.476	0.464
TD	0.603	0.405	0.432	0.254
WT	0.943	0.939	0.892	0.885

the numerical results—in the model with information about edges a smaller penetration of objects into the background can be observed.

4.2 Precision

In order to choose the correct parameters and computing precision, a software model of the Clustering algorithm was implemented in C++ with the use of the OpenCV library [30]. All calculations were performed with fixed point numbers and the precision and number of background models were adjustable (from 1 to 4).

In the first step, the influence of different fixed point precisions on the background model, particularly on calculating the running average, was examined.

$$B_{act} = \alpha_1 \cdot I + (1 - \alpha_1) \cdot B \tag{11}$$

where B background model, B_{act} updated background model, I frame and α_1 parameter controlling the background update rate.

For selected values of α_1 parameter (0.25, 0.125, 0.05, 0.01, 0.005, 0.001) and 0–5 bits for fractional part, the Eq. 11 was calculated for all possible input values (I and B in range 0–255). Then the maximal and mean error between the examined representation and double floating point precision was calculated. The results are presented in Figs. 5 and 6.

An analysis of the graphs shows that a reasonable compromise between the representation error and the number of bits allocated to the fractional part is a 3-bit precision. Assuming that the default update rate value will be $\alpha = 0.05$ relatively small errors occur. Therefore, this value was chosen as a basis for the background module.

Another issue is the representation of the gradient magnitude. The Sobel operator used on L component takes values in range [−400; 400], therefore, the magnitude (calculated according to (6)) can be in the range [0, 800]. The storing of the full information about gradients requires 10 bits for the integer part and 3 bits for the fractional part (according to the analysis made above). However, due to the auxiliary nature of the information, it was decided to limit the representation to 6 bits integer part (range 0–63), so the vertical and horizontal gradients can be in the range [−31, 31] and the potential overflow is handled by saturation.

Starting with the maximal throughput to external memory on an ML 605 platform and the input image resolution $1,920 \times 1,080$, the following parameters were chosen. The number of background models was set to $K = 3$ and the representation of a single pixel was set to:

- component L —10 bits (7 for integer part, 3 for fractional part)
- components a, b — 2×11 bits (8 for integer part, 3 for fractional part)

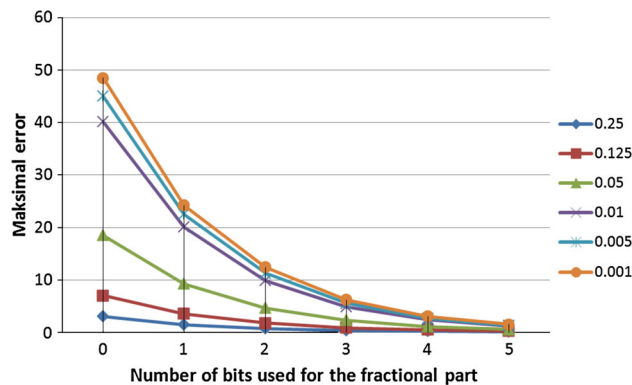


Fig. 5 The maximum representation error, depending on the number of bits for the fractional part for different values of the parameter α

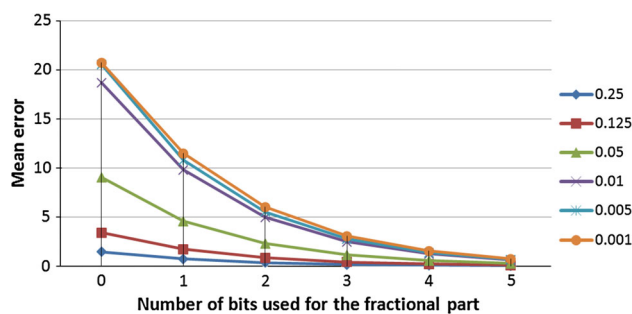


Fig. 6 The mean representation error, depending on the number of bits for the fractional part for different values of the parameter α

- gradient magnitude M —9 bits (6 for integer part, 3 for fractional part)
- weight—6 bits (parameter described in detail in Sect. 4.3)

Summing up, one model consumes 47 bits to represent a pixel of the background, three models use 141 bits.

4.3 Processing steps

When processing a video stream the following steps are carried out (independently for each pixel):

- Calculating the distance between a new pixel and each of the clusters. Distances are computed separately for luminance, chrominance and gradient magnitude based on the equations:

$$dL = |L_F - L_{Mi}| \tag{12}$$

$$dC = |Ca_F - Ca_{Mi}| + |Cb_F - Cb_{Mi}| \tag{13}$$

$$dM = |M_F - M_{Mi}| \tag{14}$$

where L_F, Ca_F, Cb_F, M_F are the values for current frame (luminance, chrominance and edge magnitude), $L_{Mi}, Ca_{Mi}, Cb_{Mi}, M_{Mi}$ values from i th background cluster.

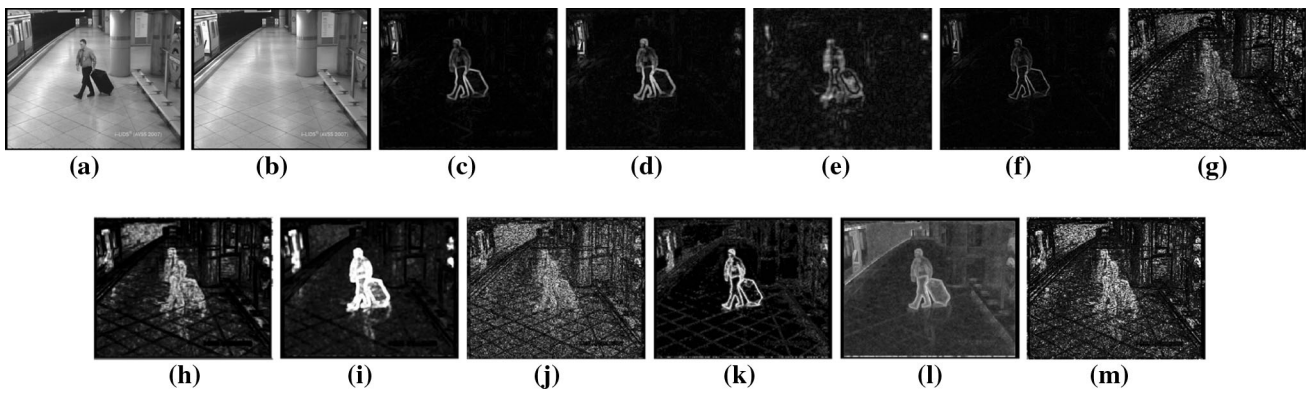


Fig. 8 The examples of the use of texture operators: **a** scene, **b** background model, **c** local standard deviation, **d** local range, **e** local entropy, **f** Sobel edge detector—magnitude, **g** Sobel edge detector—direction, **h** NCC, **i** NGD, **j** LBP, **k** SILTP, **l** RD measure and **m** measure introduced by Sanin [36]. Images originate from [15]

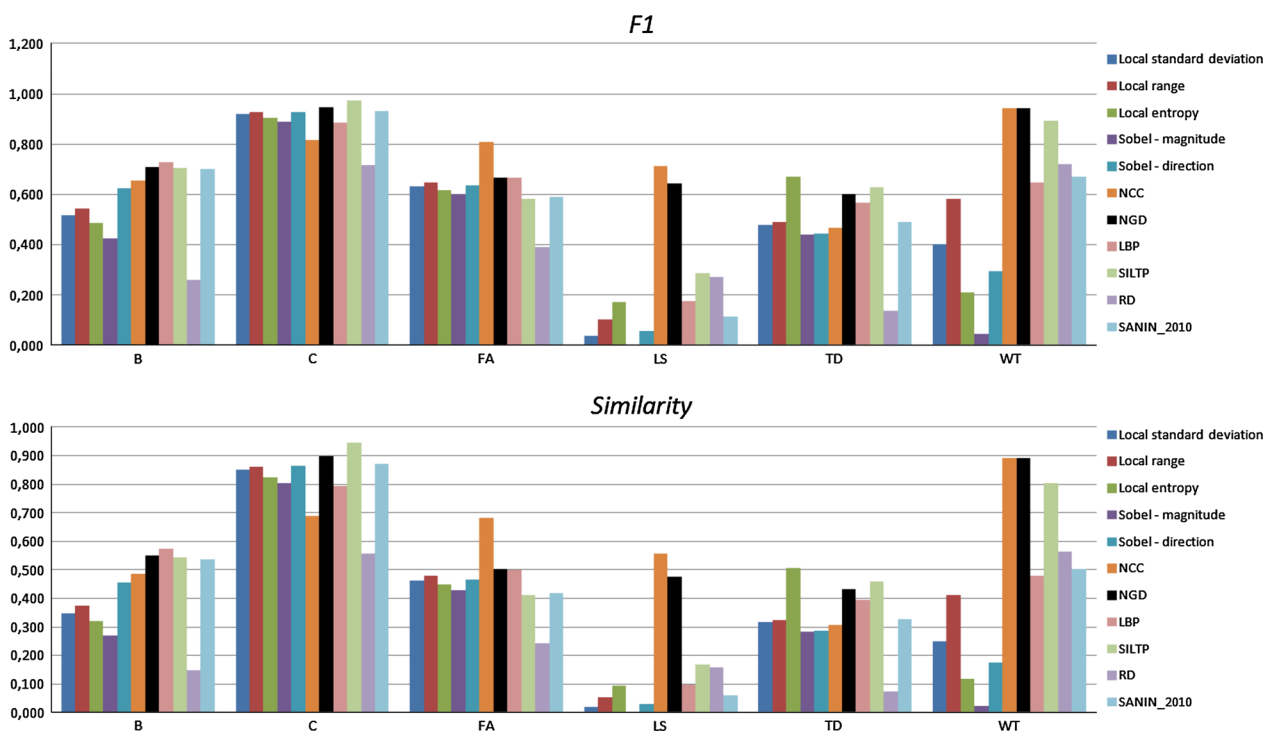


Fig. 9 Evaluation of different edge descriptors on the Wallflower dataset

Two basic properties of shadow has to be taken into consideration when designing a shadow removal method: the shadow does not change the colour but only the lightness (tests conducted in day light showed that this is not entirely true) and the shadow does not have an impact on the texture of a surface [27]. After the analysis of the results from the work [3], the CIE Lab colour space seems to be the best choice.

In order to exploit the second property, in the preliminary research phase several texture descriptors: local standard deviation, local range, local entropy, Sobel edge detector, normalized cross correlation (NCC) [17], normalized gradient difference (NGD) [23], local binary

patterns (LBP) [28], shift invariant local ternary patterns (SILTP) [33], RD measure presented in [40] and measure introduced in [36] were implemented and tested on on the Wallflower dataset. The examples of the analysed texture operators are presented in Fig. 8.

Measures *F1* and *Similarity* were computed for each descriptor for all the test sequences. The results are summarized in the graphical form in Fig. 9. The analysis of the obtained data shows that in each case the NGD descriptor score is among the three best results. None of the other descriptors is able to improve the foreground object mask so significantly. This is why the NGD descriptor was implemented in the final hardware system.

5.1 Hardware implementation

In this paper, a foreground segmentation method with shadow removal is proposed. It is based on three parameters: lightness (L component from the CIE Lab colour space), colour (a , b components) and the NGD texture descriptor. The distance of the current frame to the background is computed according to Eqs. 12 and 13. The values of all the three parameters (lightness, colour, texture) are normalised by using a method similar to one described in [24].

$$dN = \begin{cases} 1 & \text{if } d > \max(d) \cdot \beta \\ \frac{d}{\max(d)} & \text{otherwise} \end{cases} \quad (15)$$

where β is a parameter from range (0;1] (0.75 was used for the experiments), $\max(d)$ is the maximal value of measure (12), (13) and (14). Moreover, for chrominance a mechanism for removing small values (noise) was implemented.

Based on normalised values, a combination of three measures was proposed:

$$LCT = w_L dNL + w_C dN(ab) + w_T NGD_N \quad (16)$$

where w_L, w_C, w_T weights (values determined by experiments are 1, 3, 2), dNL is the normalised difference of lightness, $dN(ab)$ is the normalised difference in colour, NGD_N normalised NGD descriptor. In the last step of the algorithm, the LCT parameter is thresholded with fixed threshold (0.95 in conducted experiments). A 5×5 binary median filtering was chosen for the final image processing.

The block schematic of the segmentation module is presented in Fig. 10. It was described in VHDL language by

utilizing IP Cores generated in the Xilinx Core Generator (multiplication, delay lines). The resolution $1,920 \times 1,080$ was used for implementation. This is important, because of the delay lines length used in the SOBEL, NGD and the MEDIAN 5×5 blocks as well as the final resource utilisation and the latency introduced by the module.

Modules description:

- $\|L\|$ and $\|ab\|$ —computing distance between the current frame and the background according to Eqs. 12 and 13
- NGD—module for NGD computation,
- D—delay (for synchronization of pipelined operations),
- NORM—module for normalising the values into range [0;1],
- INTEGRATION—module for integrating lightness, colour and texture information, and the final thresholding operation (foreground/background decision),
- MEDIAN 5×5 —binary median with 5×5 window.

5.1.1 NGD computation

The normalized gradient difference (NGD) [23] is defined as:

$$R(i, j) = 1 - \frac{\sum_{(n,m)} 2 \cdot \|\nabla I(n, m)\| \cdot \|\nabla B(n, m)\| \cdot \cos(\theta)_{(n,m)}}{\sum_{(n,m)} \|\nabla I(n, m)\|^2 + \|\nabla B(n, m)\|^2} \quad (17)$$

Fig. 10 Diagram of the segmentation module

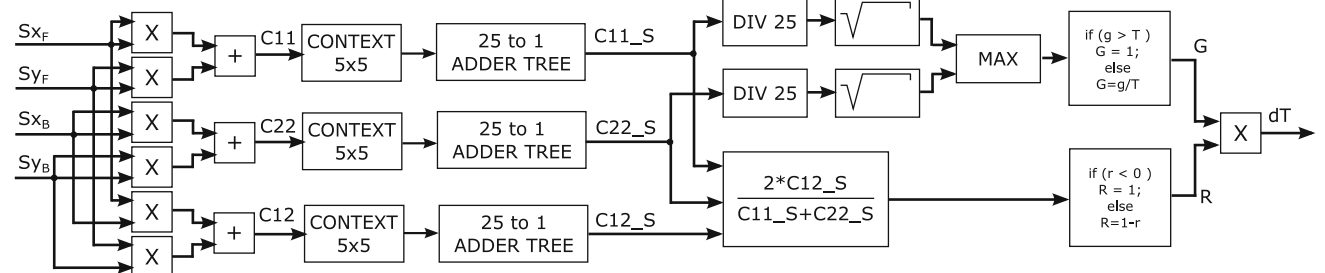
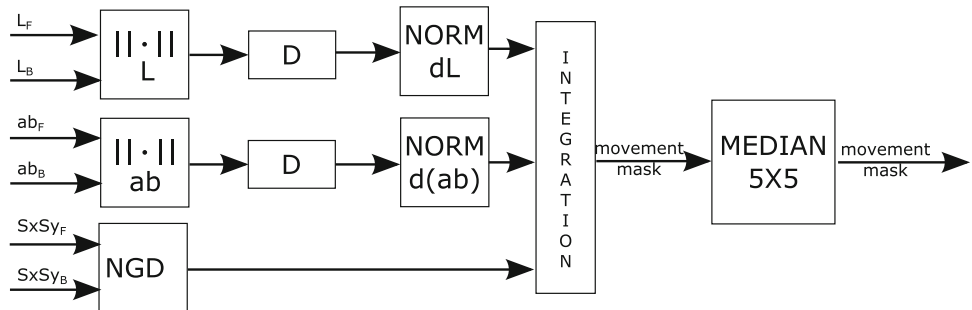


Fig. 11 Proposed architecture for NGD computation

where ∇I is the gradient (horizontal and vertical) of the current frame, ∇B is the gradient (horizontal and vertical) of the current background, $|||$ is the magnitude and θ is the angle between ∇I and ∇B .

The block schematic for the normalized gradient difference (NGD) texture descriptor computation is presented in Fig. 11. The inputs are Sobel gradients for x and y directions from the current frame and the background. Then the cross and auto correlation for the gradients are obtained by summation of multiplication results for the appropriate pairs. In the next step, the sum of the correlation parameters within the 5×5 window is determined. This is done by gathering the 5×5 context for each group, then the 25 to 1 adder tree is used to sum all the values together (designed in pipelined cascade fashion to maximize the clock speed). The accumulated results of each window are provided to the next block, which is responsible for the computing parameters G and R, using two more complex operations, namely division and square root (IP Cores provided by Xilinx are used). Finally, some thresholding operations are done on both G and R and the result is obtained by taking the dot product of them (a detailed description one can find in [23]).

6 External memory operations

The ML605 board is equipped with 64-bit data bus to DDR3 memory which is working with 400 MHz clock (data rate is 800 MHz as it is a DDR memory). The user logic is working with only 200 MHz clock, so the memory port width is 256-bit (to allow full bandwidth). The maximum theoretical data transfer for this hardware configuration can be computed as 2×400 MHz, 8 bytes (64 bits) which is 6,400 MB/s. Yet in dynamic memories not only data is transferred but also commands. Moreover the access time is not constant (depending whether a bank or column has to be opened as well as refresh commands must be issued periodically).

In the described implementation an HD video stream is processed ($1,920 \times 1,080 @ 60$ fps) therefore it was necessary to determine the maximum model width which can be used in the background generation module (Sect. 4). The data rate for an HD stream is $1,920 \times 1,080 @ 60$ fps = 124.416 MHz, yet the pixel clock is 148.5 MHz (as there are additional blanking periods). For background generation the model for a pixel has to be loaded from the memory and stored back in each clock cycle, so access to the memory with at least 248.832 MHz clock is needed. As the maximum memory bandwidth computed previously is 6,400 MB/s, the maximum theoretical memory model width is 205 bits.

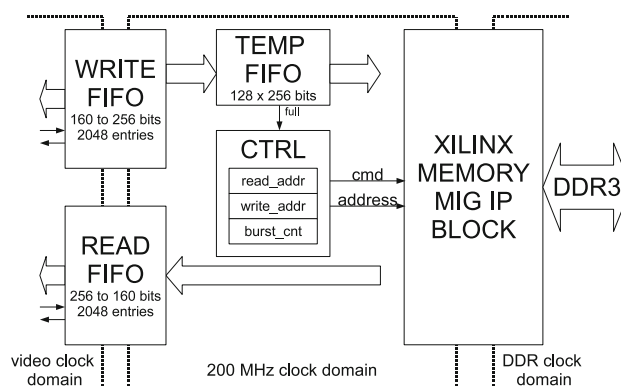


Fig. 12 RAM controller block diagram

Another problem is, that the memory interface port width is fixed and set to 256 bits. Reducing it to non power of two widths is problematic. That is why only three combinations were checked, 128-bit, 160-bit (128 + 32), 192-bit (128 + 64). During the simulation and test phase, it was possible to sustain an uninterrupted data flow for both 128-bit and 160-bit width transfers. As for the 192-bit model, it turned out that, however theoretically possible, it is not achievable (because in order to reduce the 148.5 MHz pixel clock to 124.4 MHz data clock, a very large FIFO would be needed).

Xilinx is providing an example design of a memory controller IP core (called Memory Interface Generator, MIG) for the Virtex 6 device family. It is a highly optimized design which ensures a very efficient way of communicating with external memory. It automatically calibrates, initializes and refreshes the memory, so the designer is responsible only for providing some control logic to issue the read or write commands with valid addresses and transferring the data to and from the IP core. In order to achieve maximum performance the user logic presented in Fig. 12 is proposed.

Non power of two (160–256 and 256–160 bits) FIFO’s were designed to allow data conversion between the background generation module and the memory controller as well as a clock domain crossing.

At the initialization stage the background generation module is turned off, that is why the read FIFO can be loaded with data without any interruption. After it is filled, the module waits for the vertical synchronisation signal (the moment when no video data is present, so the background generation can be safely turned on). When a new frame from the camera is transmitted, the background generation module loads a pixel model from the READ FIFO, processes it and stores it back to WRITE FIFO. From the WRITE FIFO, it is transferred to the small TEMP FIFO. When this FIFO is full, the burst sequence is triggered, storing all data from TEMP FIFO to the external memory. In the next step, the read burst sequences are

triggered to read the same amount of new pixel models. Then the module returns to the idle state (waiting for the full flag again).

The approach with TEMP FIFO is beneficial as only a full length burst access is initialized (no short bursts). Because to fill the TEMP FIFO, the exact same amount of pixel models has to be removed from READ FIFO. This means, that only by checking the full flag of the TEMP FIFO, the controller gains the information that it has both enough data to transfer as well as enough free space to store the incoming data, moreover it is a fixed and the same number of bytes.

7 Additional modules

7.1 Parameter setting (REGS, UART)

To allow parameters of the system to be changed in real time, a PL2031 USB to UART bridge available on an ML605 board was used. At the FPGA side, the UART module for transmitting and receiving data via the RS232 protocol was implemented which is able to read and write thirty-two 16-bits registers. Those registers are connected to particular module inputs which allow changes to their behaviour.

7.2 Visualisation (VIDEO OUT)

Although the ML605 board has a DVI output, it does not support the HD video stream. This is why, the video result is transmitted from the FPGA to the external HDMI encoder on the FMC module. To do this, the processed video has to be reformatted (using hardware DDR buffers) and the encoder has to be configured correctly by the I²C bus (A Picoblaze processor is applied).

8 System integration

All modules described in Sects. 3–7 were integrated according to the block diagram presented in Fig. 1. The project was synthesised for a Virtex 6 (XC6VLX 240T-

Table 3 Project resource utilisation

Resource	Used	Available	Percentage
FF	22,301	301,440	7
LUT 6	16,594	160,720	11
SLICE	6,029	37,680	16
DSP 48	35	768	4
BRAM_36	57	416	13
BRAM_18	44	832	5

1FF1156) FPGA device using Xilinx ISE 13.4 Design Suite.

Simulations performed in ModelSim 6.5c (behavioural and after place and route) confirmed that the hardware modules are fully compliant with software models described in C++. The reported maximal operating frequency (after place and route phase) was 172 MHz, which allows processing a colour HD video stream @60 frames per second. The power consumption reported by Xilinx XPower Analyzer for the device (On-Chip) is about 7.07 W. In addition, two power measurements for the whole ML605 board were performed: without running logic (14.16 W) and with running logic (24.6 W). Therefore, the FPGA system power consumption was about (10.44 W). The resource usage is presented in Table 3.

The remaining logic can be used for implementing initial image filtering (elimination of camera noise), implementing median filtering between the background generation and segmentation module or other image processing operations, except for those which need external memory access.

In Table 4, a comparison between the power consumption of the described design (Virtex 6 FPGA) and a previous version of the moving object detection system (Spartan 6 FPGA) [22] is presented. The first noticeable difference between the designs is the video stream resolution. On the SP605 board the throughput to external RAM memory is too low to support a HD stream. Furthermore the SILTP descriptor was replaced by the NGD descriptor, because the conducted research showed that it generates better result (details in Sect. 5). The power measurements indicate that the Spartan 6 design consumes approximately 8 times less power than the Virtex 6 one, but on the other hand the Virtex 6 design performs more GOPS (the method for computing parallel performance was described in [9]). However, it would be possible to implement the HD version of the algorithm on a board with Spartan 6, but only if a high throughput to external RAM was available (e.g. more than one DDR3 RAM bank).

9 Results and conclusions

9.1 Algorithm

The foreground object segmentation method proposed in this work assumes the integration of three pieces of information: lightness, colour and texture in order to obtain better results and allow the removal of shadows. It was already pointed out above that this approach gives better results than using only lightness. The results also confirm that using the colour background model gives better results (although the memory complexity is three times higher). Figure 13 presents such a situation. In Fig. 13c it can be

Table 4 Power consumption comparison

Hardware	Image resolution (@ 60 fps)	Pixel clock (MHz)	Texture descriptor	Power, W (XPower)	Power, W (measured)	GOPS	GOPS/W
Spartan 6 (SP605)	640 × 480	25	SILTP	0.9	1.2	5.45	4.54
Virtex 6 (ML605)	1,920 × 1,080	148	NGD	7.07	10.44	38.33	3.67

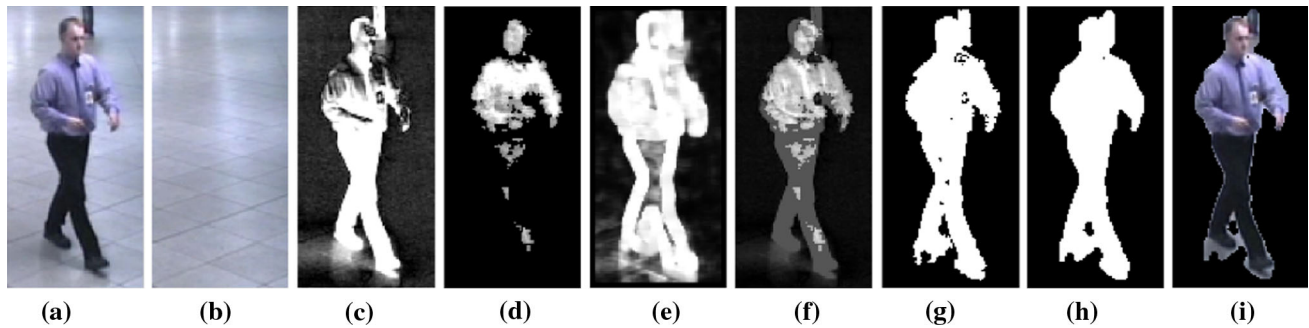


Fig. 13 Segmentation example, **a** current frame, **b** background, **c** difference in lightness, **d** difference in colour, **e** NGD texture descriptor, **f** integration of information **g**) thresholded image, **h** thresholded image with 5 × 5 median, **i** result. Images originate from iLIDS [15]

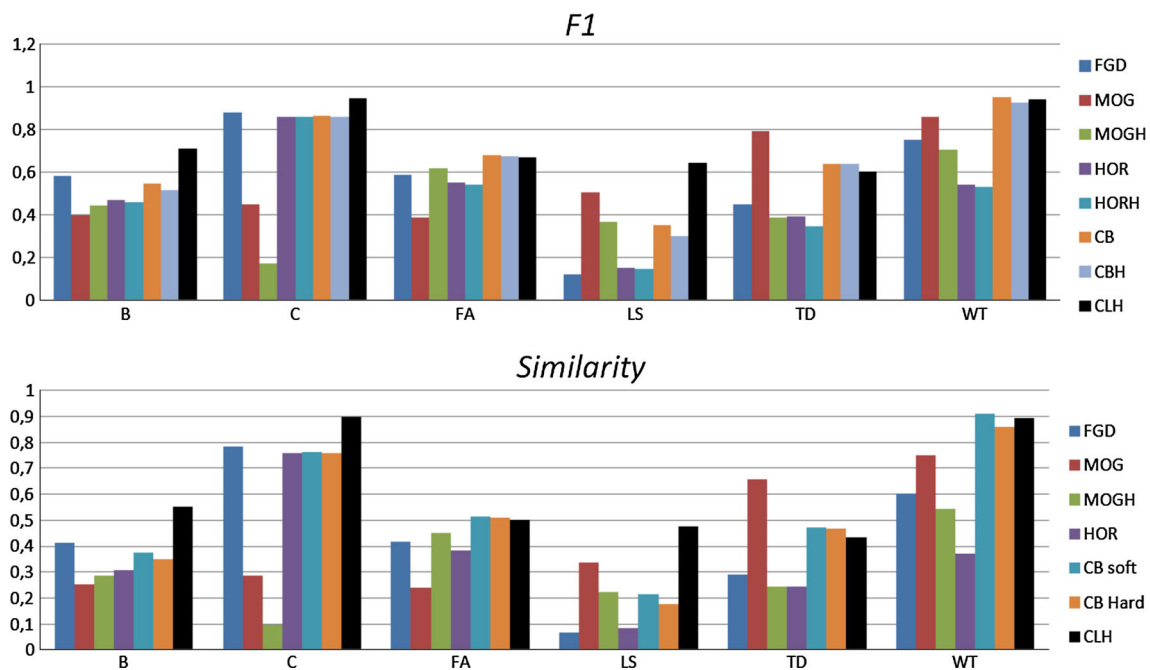


Fig. 14 The performance of the proposed algorithm evaluated using *F1* and *Similarity* measures

observed that the lightness of the persons shirt (upper part) is almost the same as the background and it is impossible to propose a good threshold for the whole silhouette. Information about the colour (Fig. 13d) allows a better segmentation. The NGD texture descriptor (Fig. 13e) provides additional information. The integration of all the features (Fig. 13f) according to the Eq. 16 allows for the proper segmentation of the silhouette (Fig. 13h).

The implemented method was also evaluated on multiple video sequences from the Wallflower [38] and Intelligent Room [32] datasets. The obtained results were compared with other papers [35] and [34]. During testing the following algorithms were compared: MOG [37], segmentation with Bayes decision rules (FGD) [24], Codebook (in original version [21] (CB) and hardware modified (CBH) [34]), simplified Multiple of Gaussian (MOGH) [2],

Fig. 15 The Wallflower test images, ground truth and segmentation results

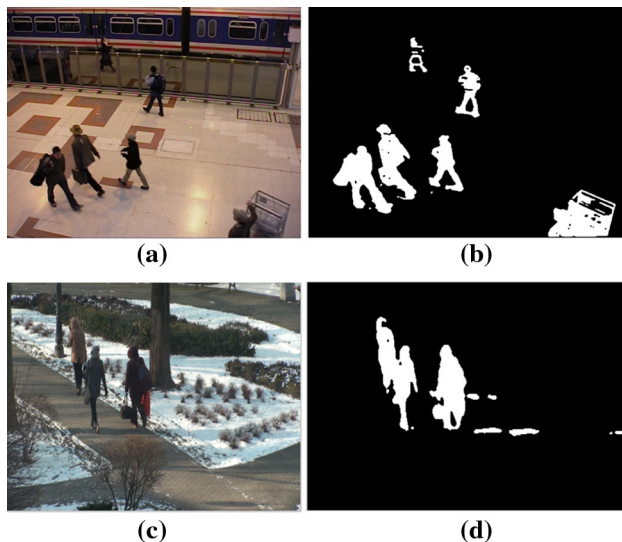
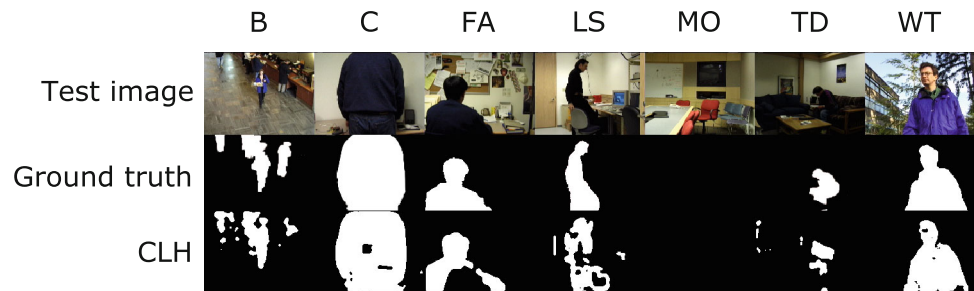


Fig. 16 Sample shadow removal. Correct removal (no strong light), a scene, b foreground object mask (binary image). Wrong removal (strong light, deep shadows) c scene, d foreground objects mask (binary image). Image a originates from PETS [31]

Horprasert algorithm [12] and the algorithm proposed in this article (CLH). The obtained results are presented as charts in Fig. 14, the algorithm proposed in this article is marked in black.

The proposed method gave the best results for the B, C, LS sequences and almost the same as the Codebook algorithm for the FA and WT sequences. Only for the TD test sequence the algorithm gave slightly worse results. Based on the mentioned comparison it can be stated that the algorithm is on the forefront of object segmentation methods. The graphical results for the Wallflower dataset are presented in Fig. 15, the examples for other algorithms can be found in [35] and [34].

The shadow removal performance is heavily influenced by using only local information (pixel, small context) and in many cases it fails. Research and literature seem to confirm this observation. However, it is possible to point out situations (Fig. 16a, b) where the proposed method is able to reduce shadow impact. In the case presented in Fig. 16c and d, with stronger light, the shadows become deeper and the proposed algorithm is not able to make

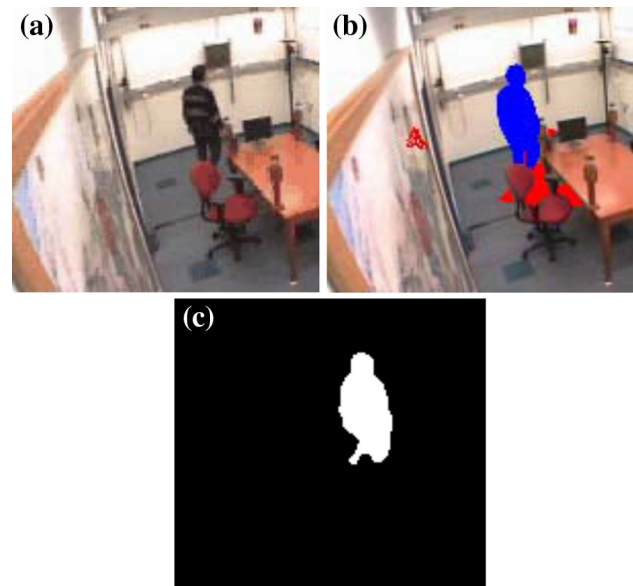


Fig. 17 Shadow elimination example. a Input image, b ground truth (blue object, red shadow), c obtained segmentation results

proper segmentation of the silhouettes. It is also worth mentioning that the described approach is less sensitive to choosing the final binarization threshold than methods using only some of information (e.g. lightness).

The “Intelligent Room” sequence [32] was used to analyse the performance of shadow removal. Because the proposed algorithm is not providing the shadow mask explicitly, it was not possible to use the methodology described in [32]. We propose a different measure. The number of pixels from shadow falsely reported as objects were counted and divided by the total size of shadow mask (result 7.6 %) and the number of pixels reported as background and truly belonging to shadow of an object (result 92.4 %). This results prove the high efficiency of the proposed algorithm in shadow removal from foreground mask. Example result is presented in Fig. 17.

9.2 System

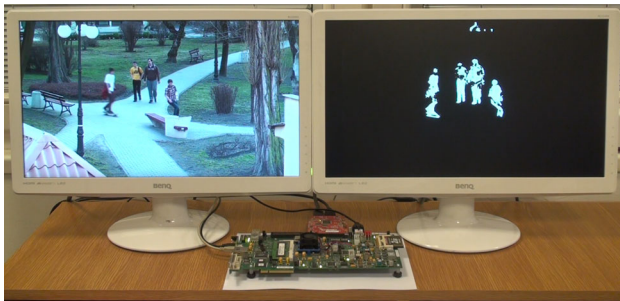
The described FPGA system for detecting foreground objects was integrated and tested in a real life environment.

Table 5 Comparison with other hardware implementations of background generation and foreground segmentation algorithms

Paper	Method	Colour	Image Resolution	Frame rate	MPPS*
[2]	mod. MOG	Y	768 × 576	145	61.18
[19]	MOG	N	1,024 × 1,024	38	38.00
[29]	Horprasert	Y	240 × 120	30	0.824
[20]	histogram	N	640 × 480	132	38.68
[10]	histogram	N	768 × 576	25	10.55
[1]	histogram	N	768 × 576	1,198 ^a	505.40 ^b
[8]	MOG	N	1,920 × 1,080	20	39.55
[35]	Horprasert	Y	1,024 × 1,024	32.8	32.8
[34]	Codebook	Y	768 × 576	50	21.09
Our approach	Clustering	Y	1,920 × 1,080	60	118.65

^a Mega pixel per second

^b External RAM operations not considered

**Fig. 18** Working system (1,920 × 1,080 @ 60 fps)

It is able to work with a targeted HD resolution (1,920 × 1,080) at 60 fps on colour images in the CIE Lab colour space. For comparison, the same algorithm implemented in C++ requires 1.7 s to process a single HD frame on a standard PC with Intel Core i7 2600 3.4 GHz processor. The estimated computational power of the presented hardware processor is 38.33 GOPS (additions, subtractions, multiplications, divisions, square root and comparisons) and the data rate between the FPGA and the external RAM is 4,976 MB/s. The module introduces a latency of over six image lines, mainly due to the three context operations: Sobel edge detection, NGD texture operator and median filtering.

The proposed hardware design was compared with hardware implementations of background generation and foreground segmentation algorithms described in literature in a similar way to the one presented in the work [35]. The results are presented in Table 5. The designed hardware module allows to process HD video stream and has the best MPPS ratio (Mega Pixels Per Second).

The system works properly and according to expectations. An example of segmentation result is presented in Fig. 18. The current video frame (left) and the foreground object mask (right) are displayed on the HD LCD monitors.

10 Summary

A system for foreground object segmentation with shadow removal implemented in an FPGA device was described in this article. It consists of several different hardware modules described in Verilog and VHDL languages: the HDMI image acquisition, the RGB to CIE Lab conversion, the Sobel edge detection, the background generation, the segmentation, the NGD texture descriptor, the external RAM controller, the serial communication with a PC and the interface to the HDMI display. A complex background generation algorithm combined with foreground object segmentation considering three features: lightness, colour and texture was implemented on a hardware platform. Finally, a real-time system, able to process 60 fps, with a resolution of 1,920 × 1,080 pixels in colour, was created. Test performed on the Wallflower data set indicate that the proposed solution is one of the best available and a comparison with other hardware implementations shows that the system offers the highest pixel per second processing rate. The results show that an FPGA device is a well suited platform for implementing sophisticated image processing algorithms for video surveillance systems and other computer vision applications.

Acknowledgments The work presented in this paper was supported by the National Science Centre of the Republic of Poland under grant no. 2011/01/N/ST7/06687 (the first author) and Ministry of Science and Higher Education of the Republic of Poland, under the project 'SIMPOZ'—grant no. 0128/R/t00/2010/12 (the second and the third author). The Authors would like to thank Enrique J. Fernandez-Sanchez from the Department of Computer Architecture and Technology, Technical School of Computer Engineering and Telecommunications, University of Granada, Spain for providing data presented in the papers [34, 35].

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

1. Abutaleb, M., Hamdy, A., Abuelwafa, M., Saad, E.: FPGA-based object-extraction based on multimodal sigma–delta background estimation. In: 2nd International Conference on Computer, Control and Communication, 2009. IC4 2009. pp 1–7 (2009). doi:10.1109/IC4.2009.4909253
2. Appiah, K., Hunter, A.: A single-chip FPGA implementation of real-time adaptive background model. In: 2005 IEEE International Conference on Field-Programmable Technology, 2005. Proceedings, pp 95–102. (2005). doi:10.1109/FPT.2005.1568531
3. Benedek, C., Szirányi, T.: Study on color space selection for detecting cast shadows in video surveillance. *Articles. Int. J. Imaging Syst. Technol.* **17**, 190–201 (2007). doi:10.1002/ima.v17:3, <http://portal.acm.org/citation.cfm?id=1298395.1298400>
4. Bouwmans, T., El Baf, F., Vachon, B.: Background modeling using mixture of gaussians for foreground detection—a survey.

- Recent Patents Comput. Sci. **1**(3), 219–237 (2008). <http://hal.archives-ouvertes.fr/hal-00338206/en/>
5. Brox, T., Bruhn, A., Papenbergh, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: European Conference on Computer Vision (ECCV), pp 25–36. Springer, Berlin (2004)
 6. Butler, D., Sridharan, S., Bove, V.: Real-time adaptive background segmentation. In: 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03), vol 3, pp III 349–352 (2003). doi:[10.1109/ICASSP.2003.1199481](https://doi.org/10.1109/ICASSP.2003.1199481)
 7. Elhabian, S.Y., Ash El-Sayed, K.M.: Moving object detection in spatial domain using background removal techniques—state-of-art. Recent Patents Comput. Sci. **1**, 32–34 (2008)
 8. Genovese, M., Napoli, E.: FPGA-based architecture for real time segmentation and denoising of HD video. J. Real-Time Image Process. **2011**, 1–13 (2011). <http://dx.doi.org/10.1007/s11554-011-0238-1>
 9. Gorgon, M.: Parallel performance of the fine-grain pipeline FPGA image processing system. Opto-Electronics Review **20**(2), 153–158 (2012). doi:[10.2478/s11772-012-0021-2](https://doi.org/10.2478/s11772-012-0021-2)
 10. Gorgon, M., Pawlik, P., Jablonski, M., Przybylo, J.: Fpga-based road traffic videodetector. In: 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, 2007. DSD 2007, pp 412–419, (2007). doi:[10.1109/DSD.2007.4341500](https://doi.org/10.1109/DSD.2007.4341500)
 11. Horn, B.K.P., Schunck, B.G.: Determining optical flow. Artif. Intell. **17**, 185–203 (1981)
 12. Horprasert, T., Harwood, D., Davis, L.S.: A Statistical Approach for Real-Time Robust Background Subtraction and Shadow Detection, pp 1–19 (1999)
 13. Huang, S.C.: An advanced motion detection algorithm with video quality analysis for video surveillance systems. IEEE Trans. Circuits Syst. Video Technol. **21**(1), 1–14 (2011)
 14. ICC: ICC.1:2004-10 specification (profile version 4.2.0.0) image technology colour management architecture, profile format, and data structure (2004)
 15. iLIDS: http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007_d.html, Access 31.01.2012 (2007)
 16. ITU-R ITU-R recommendation BT.709, basic parameter values for the HDTV standard for the studio and for international programme exchange (1990)
 17. Jacques, J., Jung, C., Musse, S.: Background subtraction and shadow detection in grayscale video sequences. In: 18th Brazilian Symposium on Computer Graphics and Image Processing, 2005. SIBGRAPI 2005, pp 189–196 (2005). doi:[10.1109/SIBGRAPI.2005.15](https://doi.org/10.1109/SIBGRAPI.2005.15)
 18. Javed, O., Shafique, K., Shah, M.: A hierarchical approach to robust background subtraction using color and gradient information. In: Workshop on Motion and Video Computing, 2002. Proceedings, pp 22–27, (2002). doi:[10.1109/MOTION.2002.1182209](https://doi.org/10.1109/MOTION.2002.1182209)
 19. Jiang, H., Ardo, H., Owall, V.: Hardware accelerator design for video segmentation with multi-modal background modelling. In: IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005, pp 1142–1145, vol. 2 (2005). doi:[10.1109/ISCAS.2005.1464795](https://doi.org/10.1109/ISCAS.2005.1464795)
 20. Juvonen, M., Coutinho, J., Luk, W.: Hardware architectures for adaptive background modelling. In: 2007 3rd Southern Conference on Programmable Logic, 2007. SPL '07, pp 149–154 (2007). doi:[10.1109/SPL.2007.371739](https://doi.org/10.1109/SPL.2007.371739)
 21. Kim, K., Chalidabhongse, T.H., Harwood, D., Davis, L.: Real-time foreground-background segmentation using codebook model. Real-Time Imaging **11**(3), 172–185 (2005)
 22. Kryjak, T., Komorkiewicz, M., Gorgon, M.: Real-time moving object detection for video surveillance system in FPGA. In: The 2011 Conference on Design & Architectures for Signal and Image Processing (DASIP), pp 209–216 (2011)
 23. Li, L., Leung, M.: Integrating intensity and texture differences for robust change detection. IEEE Trans. Image Process. **11**(2), 105–112 (2002). doi:[10.1109/83.982818](https://doi.org/10.1109/83.982818)
 24. Li, L., Huang, W., Gu, I.Y.H., Tian, Q.: Foreground object detection from videos containing complex background. In: Proceedings of the Eleventh ACM International Conference on Multimedia, MULTIMEDIA '03, pp 2–10. ACM, New York (2003)
 25. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision (darpa). In: Proceedings of the 1981 DARPA Image Understanding Workshop, pp 121–130 (1981)
 26. McFarlane, N.J.B., Schofield, C.P.: Segmentation and tracking of piglets in images. Mach. Vis. Appl. **8**, 187–193 (1995). <http://dx.doi.org/10.1007/BF01215814>
 27. Nghiem, A., Bremond, F., Thonnat, M.: Shadow removal in indoor scenes. In: IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance, 2008. AVSS '08, pp 291–298 (2008). doi:[10.1109/AVSS.2008.70](https://doi.org/10.1109/AVSS.2008.70)
 28. Ojala T, Pietikinen M, Menp T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
 29. Oliveira, J., Printes, A., Freire, R.C.S., Melcher, E., Silva, I.S.S.: FPGA architecture for static background subtraction in real time. In: Proceedings of the 19th annual symposium on Integrated circuits and systems design, SBCCI '06, pp 26–31. ACM, New York, (2006). doi: [10.1145/1150343.1150356](https://doi.org/10.1145/1150343.1150356)
 30. OpenCV: Opencv <http://opencv.willowgarage.com/wiki/>. Accessed 17.01.2012 (2012)
 31. PETS: Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance—Pets 2006 benchmark data (2006). <http://www.cvg.rdg.ac.uk/pets2006/data.html>
 32. Prati, A., Mikic, I., Trivedi, M., Cucchiara, R.: Detecting moving shadows: algorithms and evaluation. IEEE Trans. Pattern Anal. Mach. Intell. **25**(7), 918–923 (2003)
 33. Qin, R., Liao, S., Lei, Z., Li, S.: Moving cast shadow removal based on local descriptors. In: 20th International Conference on Pattern Recognition (ICPR), 2010, pp 1377–1380 (2010). doi: [10.1109/ICPR.2010.340](https://doi.org/10.1109/ICPR.2010.340)
 34. Rodriguez-Gomez, R., Fernandez-Sanchez, E., Diaz, J., Ros, E.: Codebook hardware implementation on fpga for background subtraction. J. Real-Time Image Process. 1–15 (2012)
 35. Rodriguez-Gomez, R., Fernandez-Sanchez, E.J., Diaz, J., Ros, E.: Fpga implementation for real-time background subtraction based on horprasert model. Sens. Actuators A. **12**(1), 585–611 (2012)
 36. Sanin, A., Sanderson, C., Lovell, B.: Improved shadow removal for robust person tracking in surveillance scenarios. In: 20th International Conference on Pattern Recognition (ICPR), 2010, pp 141–144 (2010). doi:[10.1109/ICPR.2010.43](https://doi.org/10.1109/ICPR.2010.43)
 37. Stauffer, C., Grimson, W.: Adaptive background mixture models for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999, vol. 2, pp. 2 (xxiii+637+663) (1999). doi:[10.1109/CVPR.1999.784637](https://doi.org/10.1109/CVPR.1999.784637)
 38. Toyama, K., Krumm, J., Brumitt, B., Meyers, B.: Wallflower: principles and practice of background maintenance. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 1, pp. 255–261 (1999)
 39. Wren, C., Azarbayejani, A., Darrell, T., Pentland, A.: Pfister: real-time tracking of the human body. IEEE Trans. Pattern Anal. Mach. Intell. **19**(7):780–785 (1997). doi:[10.1109/34.598236](https://doi.org/10.1109/34.598236)
 40. Zhang, W., Fang, X.Z., Yang, X., Wu, Q.: Moving cast shadows detection using ratio edge. IEEE Trans. Multimed. **9**(6), 1202–1214 (2007). doi:[10.1109/TMM.2007.902842](https://doi.org/10.1109/TMM.2007.902842)

Author Biographies



Tomasz Kryjak received MSc degree in Automatics and Robotics in 2008 from AGH University of Science and Technology in Krakow, Poland. From 2008 on permanent position of Research Assistant at the Institute of Automatics and Biomedical Engineering AGH-UST. His current research is focused on image processing and recognition, biometrics, reconfigurable FPGA systems, hardware algorithm acceleration and software/hardware co-design.



Mateusz Komorkiewicz received MSc degree in Automatics and Robotics in 2010 from AGH University of Science and Technology in Krakow, Poland. In the same year started PhD studies at the AGH-UST under the supervision of Prof. Marek Gorgoń. His main area of research is machine and computer vision with a special interest in accelerating vision algorithms using FPGA devices.



Marek Gorgoń received MSc degree in Electronics and Control Engineering in 1988, PhD in Automatic Control and Robotics in 1995 and DSc (habilitation) in 2007 all three from AGH University of Science and Technology in Krakow, Poland. From 1994 on permanent position at the Institute of Automatics and Biomedical Engineering AGH-UST, currently Associate Professor. His research interests include image processing, reconfigurable devices

and systems architecture, software-hardware co-design, and FPGA devices and applications. He is a Member of IEEE Computer Society from 2002, and a Member of IEEE from 2006. He is a Member of International Program Committees of ACM-SAC Symposium, ERSA Conference, ReConFig Conference and Design and Architectures for Signal and Image Processing. He is an author of 70 technical papers and reports. He was an associate editor of Proceedings of International Conference on Engineering of Reconfigurable System and Algorithms (ERSA) in 2002. In 2002, 2004, 2008 and 2011 he obtained the awards of the Rector of AGH-UST University. He participated in 11 scientific and industrial research projects. He regularly reviews for International journals, conferences, and State Committee for Scientific Research.