Real-time Foreground Object Detection Combining the PBAS Background Modelling Algorithm and Feedback...

Article *in* International Journal of Electronics and Telecommunications · March 2014

DOI: 10.2478/eletel-2014-0006

Tomasz Kryjak
AGH University of Science and Technology in …
60 PUBLICATIONS **123** CITATIONS

M. Gorgon
AGH University of Science and Technology in …
70 PUBLICATIONS **219** CITATIONS

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2014, VOL. 60, NO. 1, PP. 61–72
Manuscript received February 10, 2014; revised March, 2014. DOI: 10.2478/eletel-2014-0006

Real-time Foreground Object Detection Combining the PBAS Background Modelling Algorithm and Feedback from Scene Analysis Module

Tomasz Kryjak, Mateusz Komorkiewicz, and Marek Gorgon

***Abstract*—The article presents a hardware implementation of the foreground object detection algorithm PBAS (Pixel-Based Adaptive Segmenter) with a scene analysis module. A mechanism for static object detection is proposed, which is based on consecutive frame differencing. The method allows to distinguish stopped foreground objects (e.g. a car at the intersection, abandoned luggage) from false detections (so-called ghosts) using edge similarity. The improved algorithm was compared with the original version on popular test sequences from the `changedetection.net` dataset. The obtained results indicate that the proposed approach allows to improve the performance of the method for sequences with the stopped objects. The algorithm has been implemented and successfully verified on a hardware platform with Virtex 7 FPGA device. The PBAS segmentation, consecutive frame differencing, Sobel edge detection and advanced one-pass connected component analysis modules were designed. The system is capable of processing 50 frames with a resolution of 720 $\times$ 576 pixels per second.**

***Keywords*—PBAS algorithm, foreground segmentation, foreground object detection, background generation, background subtraction, background modelling, image processing and analysis, FPGA, connected component analysis, consecutive frame differencing**

I. Introduction

ROBUST foreground object detection and background generation are important components of many real world image processing and analysis systems. Good examples are: automated advanced video surveillance systems, video based remote patient monitoring, video traffic monitoring, autonomous guided vehicles or semantic image analysis.

The purpose of foreground object segmentation is to assign every pixel in the image into two categories: background or foreground, where by foreground are meant objects which are interesting for a given system, usually people or cars. It should be noted that this is not a simple moving or static classification. For example, a person or a car may stop for a period of time and yet should still be detected. On the other hand, one can indicate a number of situations, where the background elements are also moving. These are flowing water, fountains, moving leaves or branches. Unfortunately, they are usually detected as foreground objects by the basic segmentation methods. The elimination of this type of false detections is one of the tasks for the algorithm designer.

The work presented in this paper was supported by AGH Univeristy of Science and Technology projects number 15.11.120.330 (first author), 15.11.120.356 (second author), 11.11.120.612 (third author).

T. Kryjak, M. Komorkiewicz, and M. Gorgon are with the Department of Automatics and Biomedical Engineering, AGH Univeristy of Science and Technology, Krakow, Poland, (e-mails: {kryjak; komorkie; mago}@agh.edu.pl).

Another challenge are virtual objects (referred to in the literature as “ghost”), which are false detections present on the object mask – they do not correspond to any real objects on the current scene. They arise in two cases: when an initially stationary object begins to move (e.g. a car will leave a parking space) or when an object, that was static for some time and was introduced into the background model, starts moving again. The greatest difficulty is to distinguish ghosts from “real” stopped objects, because they have very similar properties. In both cases there is a significant difference between the current frame and the background model and also the object remains stationary.

Shadows cast by foreground object are a further difficulty. On the one hand, they have all the features of an object (they differ from the background and move), but usually they should be regarded a distortion, as they significantly influence the foreground mask’s shape. For example they can lead to “joining” two distant objects. It is also difficult to consider this type of shadows as part of the background. Therefore, it seems necessary to treat them as a separate category, in addition to the foreground and background. Furthermore, the used background model should adapt to the current lighting conditions and be somewhat resistant to the effects of camera jitter (vibrations) and unfavourable meteorological conditions (fog, snow, rain etc.).

The examples mentioned above show that the considered problem is quite complex and constant research in the field of background generation and foreground object segmentation is still needed. In the literature many methods are described. From most basic ones, based on computing the running average or mean/median from N last buffered frames, to more complex, so called multi-variant: GMM (Gaussian Mixture Models), Clustering, Codebook or eigenbackground based. Comprehensive surveys can be found in [1], [2] and [3].

In this paper we address the problem of proper handling of stopped objects and ghosts. The integration of the foreground object mask obtained by the PBAS (Pixel-Based Adaptive Segmenter) method [4] and movement mask determined using consecutive frame differencing combined with the analysis of stationarity and edge similarity is presented. The proposed system was realised in a FPGA device, which is a proven platform for hardware implementation of image processing and analysis algorithms [5]. The use of fine grain parallelism available on the hardware platform allowed to implement the PBAS, consecutive frame differencing, Sobel edge detection, median filtering and an advanced one-pass connected component analysis algorithms and create a real-time vision system able to process 50 frames with resolution 720×576 pixels per second.

The main contributions of the paper are threefold:

- an improvement of the PBAS algorithm that allows to distinguish stopped objects from ghosts, which improves the foreground object detection results,
- hardware implementation of an advanced video processing system in a low power FPGA device,
- verification of the proposed architecture on an evaluation board with connected HDMI camera on real life sequences.

In Section II the PBAS method, which is an essential component of the solution, is described. Its hardware implementation was previously reported in the work [6]. Section III depicts the evaluation methodology of foreground object segmentation algorithms, as well as the obtained results. They were the basis for the proposed modifications i.e. using motion detection and object analysis, which are presented in Section IV. Issues related to hardware implementation are discussed in Section V. First, a survey of background modelling algorithms implemented in hardware is presented – Subsection V-A. Then, in Subsection V-B the realised system is described and the essential modules: PBAS algorithm (Subsection V-C) and one-pass connected components analysis (Subsection V-D) are discussed in details. The system integration on an evaluation board with FPGA device, as well as resource utilization, energy consumption and computing performance are presented in Subsection V-E. The article ends with a summary and discussion of the possible directions of further development.

II. The PBAS Algorithm

The background model in the PBAS [4] is based on a buffer of N samples from the analysed video sequence. Let x_i denote a particular pixel. The foreground/background classification is made on the basis of comparing the model $B(x_i)$ with the current frame $I(x_i)$. For each pixel a unique decision threshold is used $R(x_i)$. The update is performed for random samples with a probability specified by the parameter $T(x_i)$. Both $R(x_i)$, $T(x_i)$ are adjusted separately for each pixel, which distinguishes PBAS from other methods. In the following section the algorithm is described in detail.

A. Segmentation

In PBAS, similarly to other foreground segmentation algorithms, the foreground/background classification is based on comparison of the model with the current pixel. In the discussed method, the background model is defined as an array of N recently observed pixel values:

$$B(x_i) = \{B_1(x_i), \dots, B_k(x_i), \dots, B_N(x_i)\} \tag{1}$$

A pixel x_i belongs to the background when the distances between the current value $I(x_i)$ and at least $\#_{min}$ samples from the background model are smaller than the threshold $R(x_i)$. Therefore the object mask is calculated as:

$$F(x_i) = \begin{cases} 1 & \text{if } \sum_{k=0}^{N} \{dist(I(x_i), B_k(x_i)) \\ & \qquad < R(x_i)\} < \#_{min} \\ 0 & \text{else} \end{cases} \tag{2}$$

where: $F = 1$ denotes foreground, $F = 0$ background and $dist$ is a distance measure:

$$dist(I(x_i), B_k(x_i)) = |I(x_i) - B_k(x_i)| \tag{3}$$

B. Background Model Update

The background model update is necessary to compensate for slight changes in lighting (e.g. time of the day), as well as some inevitable changes in the scene (motion in the background – moving trees, flowing water and the appearance and disappearance of objects – e.g. cars in the parking lot).

In a background generation methods one can distinguish two update polices: liberal and conservative. In the first case, all pixels are updated, in the second only those classified as background. Both approaches have certain unique properties. The main disadvantage of the liberal policy is a relatively rapid inclusion of foreground objects into the background model, which leads to segmentation errors (i.e. false negative errors at first and then possibly "ghosts"). In particular this applies to objects that move very slow. The conservative method avoids the phenomenon described above, however, has one major drawback. The use of the foreground object mask as an update condition leads to "deadlocks" caused, both by minor segmentation errors and motion of objects, which were initially static. Furthermore, once present an error will never be eliminated. For this reason some mechanisms to prevent such cases are designed. One possible solution is counting the number of times a pixel is classified as an object. When the counter value exceeds a threshold, an update is forced. In PBAS a conservative policy with an additional mechanism for updating the neighbouring pixels, which prevents from irreparable segmentation errors, is used.

Only pixel classified as background may be updated ($F(x) = 0$). The actualisation is based on replacing a randomly selected sample from the background model ($B_k(x_i)$) with the current pixel value ($I(x_i)$). The probability of an update is given as $p = 1/T(x_i)$. Similarly, a randomly selected sample, from a randomly selected model, from a 3×3 local context is replaced by the current value $I(y_i)$, where y_i denotes a pixel from the context.

C. Update of the Decision Threshold $R(x_i)$

In PBAS the following update mechanism for the $R(x_i)$ value is proposed. Primary, the minimum distances between samples from the model, and current pixel are defined: $D = \{D_1(x_i), \dots, D_N(x_i)\}$. If a pixel is updated the minimal distance:

$$d_{min}(x_i) = min_k dist(dist(I(x_i), B_k(x_i))) \tag{4}$$

is saved as $D_k(x_i) = d_{min}(x_i)$. The mean value of $D(x_i)$, refereed to as $\bar{d}_{min}(x_i)$, is a measure of background dynamics and is used in the $R(x_i)$ update:

$$R(x_i) = \begin{cases} R(x_i)(1 - R_{inc/dec}) & \text{if } R(x_i) > \bar{d}_{min}(x_i)R_{sc} \\ R(x_i)(1 + R_{inc/dec}) & \text{else} \end{cases} \quad (5)$$

where: $R_{inc/dec}$ – constant update rate = 0.05 [1], R_{sc} – scaling factor = 5. Additionally the decision threshold was limited by a lower bound $R_{lower} = 18$.

D. Update of the Learning Rate $T(x_i)$

The learning rate update procedure is described as:

$$T(x_i) = \begin{cases} T(x_i) + \frac{T_{inc}}{d_{min}(x_i)} & \text{if } F(x_i) = 1 \\ T(x_i) - \frac{T_{dec}}{d_{min}(x_i)} & \text{if } F(x_i) = 0 \end{cases} \quad (6)$$

where: $T_{inc} = 1$ and $T_{dec} = 0.05$ – update rates. Additionally the learning rate is limited by a lower bound $T_{lower} = 2$ and upper bound $T_{upper} = 200$. Details on the update mechanism are described in [4].

E. Additional Information

In practice, video sequences are in the RGB colour space. The authors of the paper [4] therefore proposed to process each colour component separately and to combine the resulting segmentation masks with the OR operator (i.e. $F(x_i) = F_R(x_i)$ OR $F_G(x_i)$ OR $F_B(x_i)$). As a post-processing operator median filtering with a window size of 9×9 pixels was proposed.

In the original algorithm, the background model was supplemented with information about the edges (using Sobel gradient magnitudes). However, analysing the results described in [4] lead to the conclusion that the addition of edges allowed only for slight improvement in segmentation, but increased the size of the background model and complicated the calculation of distances between the current pixel and the background model. Therefore, in the described hardware implementation this mechanism is not used.

III. Evaluation of the PBAS Algorithm

This section presents the evaluation methodology of foreground object segmentation algorithms, as well as the used test database. Furthermore, results of the basic PBAS method are discussed in details.

A. Evaluation Methodology and Test Sequences

The most common method of assessing the effectiveness of foreground object segmentation algorithms is to compare the obtained mask with a reference one (created by manual annotation) at single pixel level. One popular test datasets is the IEEE Workshop on Change Detection database [7] – `changedetection.net`. The dataset contains sequences divided into six categories (2012 version): 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 noticed that the database contains sequences which cover a large part of the situations that are problematic to background generation algorithms.

[1]default values of all algorithm parameters were proposed in the paper [4]

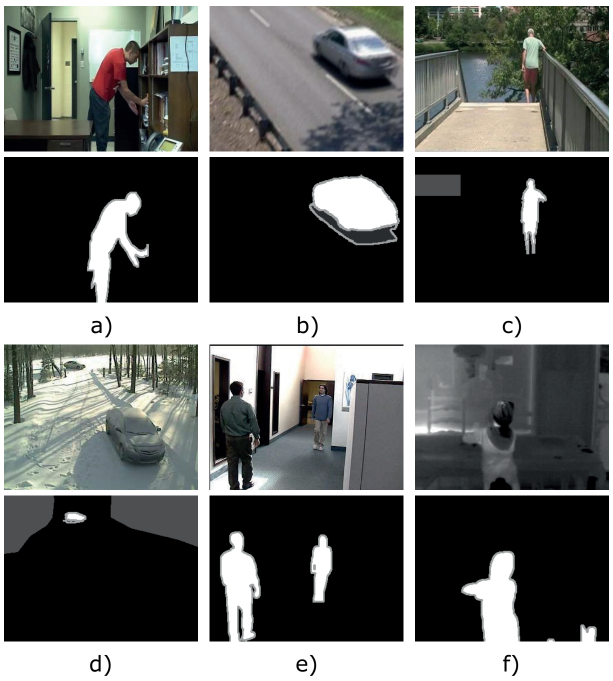

Fig. 1. Sample images and ground truths for sequences: a – *baseline/office*, b – *cameraJitter/traffic*, c – *dynamicBackground/overpass*, d – *intermittentObjectMotion/winterDrive*, e – *shadow/backdoor*, f – *thermal/diningRoom*.

However, the main advantage of the database, and what distinguishes it from other collections (e.g. Wallflower [8]), is a large number of manually annotated reference frames with areas marked as: background, shadow, movement, slight blurring and foreground. This allows for a reliable assessment of the algorithms performance in different situations. Sample images and ground truths are presented in Fig. 1.

The methodology used in the experiments can be described as follows. The object mask computed by the algorithm is compared with the reference mask. In this research only the foreground and background classification was considered, as no shadow detection procedure was implemented. The rates listed below 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 the following measures were determined:

- *Recall* = TP / (TP + FN),
- *Specificity* = TN / (TN + FP),
- *FPR* (False Positive Rate) = FP / (FP + TN),
- *FNR* (False Negative Rate) = FN / (TP + FN),
- *PWC* (Percentage of Wrong Classifications) = 100 $\times$ (FN + FP) / (TP + FN + FP + TN),
- *F1* = (2 $\times$ Precision $\times$ Recall) / (Precision + Recall)
- *Precision* = TP / (TP + FP).

B. Results of the PBAS Method

Table I summarizes the results obtained for the following algorithms: hardware implementation of the PBAS method (PBAS_FPGA) prosed in the work [6], the original PBAS method described in the article [4], the frequently used Gaussian Mixture Models (GMM) method [9] (the implementation available in the OpenCV library [10]) and Spectral-360, which is the best algorithm in the `changedetection.net` comparison (version 2012).

Analysis of the results leads to the following conclusions. The *Recall*, *Specificity* and *FNR* are quite similar for methods PBAS and Spectral-360. The PBAS_FPGA hardware implementation has a slightly higher false positive rate (*FPR*). The percent of wrong classification value (*PWC*) for the FPGA version of PBAS is higher than for the original software version and Spectral-360 method, but similar to GMM. This is the result of a higher false positive rate, which is confirmed by the *F1* and *Precision* values. The differences between the hardware and software version result from: lower number of samples in background model ($N_{FPGA} = 19$, $N_{SOFT} = 35$) and fixed point calculations (particularly the $R(x_i)$ and $T(x_i)$ rates).

In the next stage of the research the cause of errors for test sequences were examined. Table II summarizes the average parameter values in the individual categories obtained by the PBAS_FPGA method. For the *Baseline* set the results were relatively correct. In the case of sequences containing camera jitter (*Camera Jitter*) characteristic is the low value of *Precision* and *F1*. This is due to a significant number of false detections caused by camera movement. Their reduction would be possible through the use of vibration compensation mechanism based on camera movement estimation between consecutive frames (e.g. using optical flow).

Another difficult case is the presence of motion in the background (*Dynamic Background*). It is worth to distinguish between the two situations. In one, the movement is present in a location where there are no foreground objects or they appear only temporarily. An example is presented in Fig. 2a. There, the objects appear on the road behind the fountain, in front of the building. In the second, the objects occur at the same location as the moving background – a good example is a boat on the water (Fig. 2b). In the first of the cases, a fairly good solution seems to be the exclusion of the problematic areas from analysis. This can be done manually, for example by defining masks on the setup/configuration stage or automatically, using the motion history image (MHI) or blinking pixels detection mechanism described in papers [11] and [12]. A similar approach in other situations usually results in omitting large parts of foreground objects.

The correct segmentation of stopped foreground objects, as well as ghost elimination is tested on the *Intermittent Object Motion* set. In this case, the major difficulty is to propose a solution, which on the one hand will not cause the intrusion of static objects from the foreground to the background model (e.g. a person standing at a pedestrian crossing or an abandoned luggage) and on the other hand will tend to eliminate detections of "empty space" or "ghosts" (e.g. when a parked car left a parking place). The PBAS_FPGA method

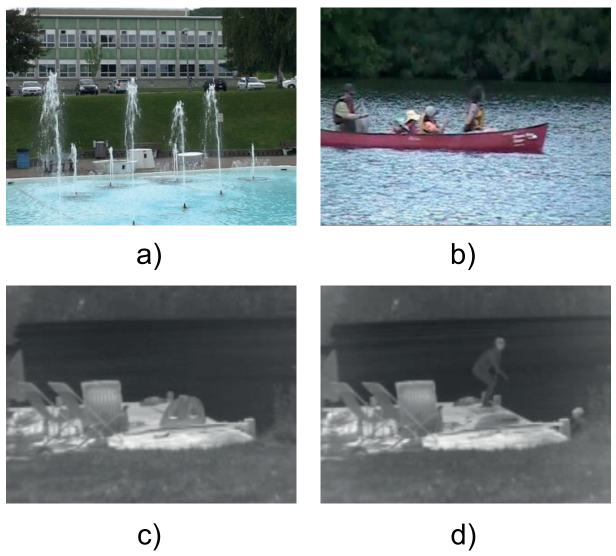

Fig. 2. Difficult sequences from the `changedetection.net` dataset: (a) fountain, (b) river, (c) and (d) thermal image without and with foreground object respectively.

achieved for these sequences rather average results, because the algorithm uses a conservative update approach combined with diffusion of samples to neighbouring background models (compare Section II-B). This results in a slowly intrusion into the background model both ghosts and foreground objects.

The results for the *Shadows* set should be considered quite decent. It is worth remembering that the PBAS algorithm does not include a mechanism to detect and eliminate shadows. Its implementation could improve the effectiveness of the method. For the *Thermal* category the results are also quite good. In this case, the main problem is the poor contrast between the objects and the background. Therefore some objects are not detected at all. It is particularly well illustrated by the sequence *Lake Side* – compare Fig. 2c, where the scene is empty and Fig. 2d with a foreground object.

IV. Improvement of the PBAS Algorithm

Based on the analysis described in Section III-B it was decided to focus on the problem associated with the *Intermittent Object Motion* dataset because of its significance for video surveillance systems, especially those whose main application is abandoned objects detection. It was pre-assumed that the algorithm should be able to work in a streaming video system realized on a platform with an FPGA device. This excluded a number of solutions, which implementation in hardware is difficult or even impossible – for example, segmentation by division or region growing (i.e. recursive approach).

The proposed solution is based on two observations:

- both the stopped foreground object and the ghost are static, i.e. not motion should be detected for a longer period of time at this area,
- the stopped object differs from the ghost in that the object "exists" in the current frame and the ghost does not.

In order to determine whether the object is static, a mechanism based on subtraction of two consecutive frames was

TABLE I
RESULTS OF PBAS AND OTHER STATE-OF-THE-ART METHODS

Method	Recall	Specificity	FPR	FNR	PWC	F1	Precision
PBAS_FPGA (N=19, RGB) – our FPGA	0.7977	0.9720	0.0280	0.2023	3.4353	0.6813	0.6994
PBAS (N=35, RGB) reported in [4]	0.7840	0.9898	0.0102	0.2160 [2]	1.7693	0.7532	0.8160
GMM (included in OpenCV) [3]	0.6964	0.9845	0.0155	0.3036	3.1504	0.6596	0.7079
Spectral-360 [4]	0.7770	0.9920	0.0080	0.2230	0.8516	0.7770	0.8461

[2] result reported for the PBAS method at changedetection.net
[3] results from the changedetection.net website
[4] results from the changedetection.net website

TABLE II
RESULTS OF PBAS_FPGA METHOD ON DIFFERENT CATEGORIES FROM THE CHANGEDETECTION.NET DATASET

Category	Recall	Specificity	FPR	FNR	PWC	F1	Precision
Baseline	0.9259	0.9968	0.0032	0.0741	0.5588	0.9179	0.9104
Camera Jitter	0.8566	0.9338	0.0662	0.1434	6.8527	0.5284	0.3897
Dynamic Background	0.8708	0.9859	0.0141	0.1292	1.5243	0.6674	0.6728
Intermittent Object Motion	0.6046	0.9356	0.0644	0.3954	7.9660	0.4893	0.5884
Shadows	0.9163	0.9848	0.0152	0.0837	1.7977	0.8016	0.7327
Thermal	0.6499	0.9945	0.0055	0.3501	1.8239	0.7173	0.9037

used. The difference in the RGB colour space is described by the formula:

$$dF(x_i) = \sum_{C\in\{R,G,B\}} |I(x_i)_K^C - I(x_i)_{K-1}^C| \qquad (7)$$

where: $I(x_i)_K^C$ denotes a colour component $C \in \{R, G, B\}$ for frame K.

As a stability measure the ratio of pixel number, for which the calculated $dF(x_i)$ value (Equation (7)) exceeds a given threshold θ to the object area was used:

$$S_{Ok} = \frac{\sum_{x_i\in O_k} dF(x_i) > \theta}{\sum_{x_i\in O_k} F(x_i)} \qquad (8)$$

where: O_k – denotes the k-th object, $F(x_i)$ – foreground object mask.

The object is considered as static if the measure S_{Ok} exceeds a low threshold S_{TH} (0.05 - 0.1). It is used to eliminate small distortions resulting mainly from the camera noise or image compression artefacts.

To determine whether the object exists in the current frame the mechanism described in the work [13] was adapted. It involves a parallel edge analysis of the object (object's mask), the current frame and the background model. The authors assumed that if the edges layout of the mask is more similar to the current frame the considered object is static. In contrary, when it is close to the background, then the object does not exist (it is a "ghost"). Due to the difficulty of obtaining a direct representation of the background in the PBAS method, it was decided to use only the information from the current frame and the object mask. In the solution, the Sobel edge detector was used. For the current frame, the edge detection was performed separately for RGB components and the results were combined with an "OR" operator. Then, the edge map was binarized with a fixed threshold and for each object a coefficient was calculated:

$$EC_{Ok} = \frac{\sum_{x_i\in O_k} F_E(x_i) == 1 \wedge I_E(x_i) == 1}{\sum_{x_i\in O_k} F_E(x_i) == 1} \qquad (9)$$

where: F_E – foreground object mask edges, I_E – current frame edges, O_k – the k-th object.

An object for which the EC_{Ok} value exceeded 0.5 was considered as present in the current frame. The threshold was chosen experimentally after a careful evaluation of several test sequences.

Using both described mechanisms and the PBAS foreground object segmentation method, the following algorithm was proposed (schematically in Fig. 3).

In the first step the object mask segmentation (PBAS), edge detection (EDGE) and the consecutive frame differencing

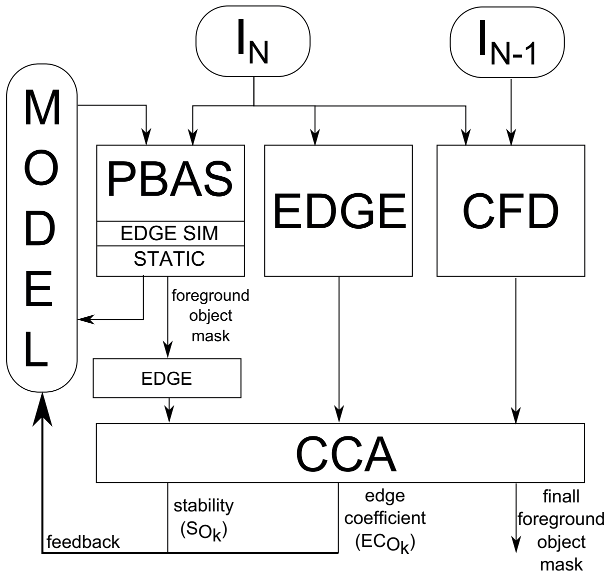

Fig. 3. Block scheme of the proposed algorithm. I_N and I_{N-1} – two consecutive frames from the video sequence, CFD – consecutive frame differencing, CCA – connected component analysis.

TABLE III
COMPARISON OF THE PBAS_FPGA AND THE IMPROVED PBAS_FPGA+ METHOD. SEQUENCES FROM THE *Intermittent Object Motion* SET

Sequence	Algorithm	Recall	Specificity	FPR	FNR	PWC	F1	Precision
Abbadoned Box	PBAS_FPGA	0.9181	0.9575	0.0425	0.0819	4.4430	0.6652	0.5216
	PBAS_FPGA+	0.9909	0.9728	0.0272	0.0091	2.6357	0.7834	0.6477
Parking	PBAS_FPGA	0.1153	0.9980	0.0020	0.8847	7.0267	0.2026	0.8320
	PBAS_FPGA+	0.5584	0.9832	0.0168	0.4416	4.9708	0.6351	0.7361
Sofa	PBAS_FPGA	0.5539	0.9962	0.0038	0.4461	2.3118	0.6767	0.8694
	PBAS_FPGA+	0.5961	0.9962	0.0038	0.4039	2.1262	0.7101	0.8782
Street Light	PBAS_FPGA	0.5278	0.9926	0.0074	0.4722	2.9979	0.6308	0.7839
	PBAS_FPGA+	0.9619	0.9999	0.0001	0.0381	0.1980	0.9792	0.9972
Tram Stop	PBAS_FPGA	0.8240	0.6992	0.3008	0.1760	27.8423	0.5151	0.3747
	PBAS_FPGA+	0.9560	0.9709	0.0291	0.0440	3.1727	0.9154	0.8780
Winter Drive	PBAS_FPGA	0.6883	0.9704	0.0296	0.3117	3.1745	0.2451	0.1491
	PBAS_FPGA+	0.7220	0.9939	0.0061	0.2780	0.8108	0.5716	0.4731

(CFD) is performed. Then, connected component analysis is realized (CCA). For each object the following data is collected:

- bounding box,
- area,
- number of pixels, for which the stability measure exceeds a given threshold $S_{Ok} > S_{TH}$,
- number of pixels, for which there is a match between the mask edges and current frame edges, as well as the number of edge mask pixels – EC_{Ok}.

It should be noted that the solution uses a single-pass connected component analysis approach, because it is much simpler and faster in an FPGA hardware implementation [14]. However, it also involves some significant limitations. At its output no labelled mask is available, but only object parameters i.e. bounding box, area, etc. Therefore, the basic representation on which the algorithm operates is a bounding box and not the object mask. The use of a typical two-pass connected component labelling could be a possible further development of the system. On the basis of the data collected for the objects the following operations are performed: small objects are excluded from analysis (with area less than a given threshold) and for others the stability (see Equation (8)) coefficient as well as the edge similarity (Equation (9)) is calculated. These results are passed to the next iteration of the segmentation algorithm and should be considered as feedback from the analysis module. In Fig. 3 it is illustrated as STATIC and EDGE SIM sub modules in the PBAS module.

Two variables were added to the PBAS background model (see Section II-A) to realize the desired functionality. The first stores the information about how many times a pixel has been recognized as a part of a stationary object $S(x_i)_{CNT}$. It is determined according to:

$$S(x_i)_{CNT} = \begin{cases} S(x_i)_{CNT} + 1 & \text{if} S_{Ok} >= S_{TH} \\ 0 & \text{if} S_{Ok} < S_{TH} \\ S(x_i)_{CNT} - 1 & \text{else} \end{cases} \tag{10}$$

The update of $S(x_i)_{CNT}$ is performed for all pixels inside a rectangular area, as a single object is represented by its bounding box. For the pixels belonging to a stationary object the value of $S(x_i)_{CNT}$ is increased, for moving ones it is set to zero and for background it is decreased.

The second variable is the running average of the edge similarity value ($EC(x_i)_{mean}$), which is updated according to the equation:

$$EC(x_i)_{mean} = \begin{cases} 0.5EC(x_i) + 0.5EC(x_i)_{mean} & \text{if A} \\ 1 & \text{else} \end{cases} \tag{11}$$

where: $A = S_{Ok} >= S_{TH} \wedge S(x_i)_{CNT} > S_{CNT_TH}$, S_{CNT_TH} is the minimal number of frames during which an object must remain static in order to perform an analysis whether it is as ghost.

In addition, the approach of neighbouring background model update in PBAS was disabled, since it causes a slowly and gradually introduction them into the model. Then, the resulting change in the object's shape makes the described previously edge analysis unfounded in this case. In return, an update mechanism for pixels considered as part of stationary objects was added. It is realised when two conditions are met: $S_{Ok} >= S_{TH} \wedge S(x_i)_{CNT} > S_{CNT_TH} \wedge EC(x_i)_{mean} < 0.5$. The update itself is performed in a manner analogous to that described in Section II-B.

The task of the above-described mechanisms is to ensure that the inclusion of a foreground object into the background model is only possible when the object is static for a long period of time and its edge layout is not similar to the one present on the current frame. Furthermore, the proposed running average mechanism "filters out" transient errors.

A. Evaluation of the Proposed Modifications

The proposed mechanism for ghost elimination and proper stopped object segmentation was evaluated on sequences

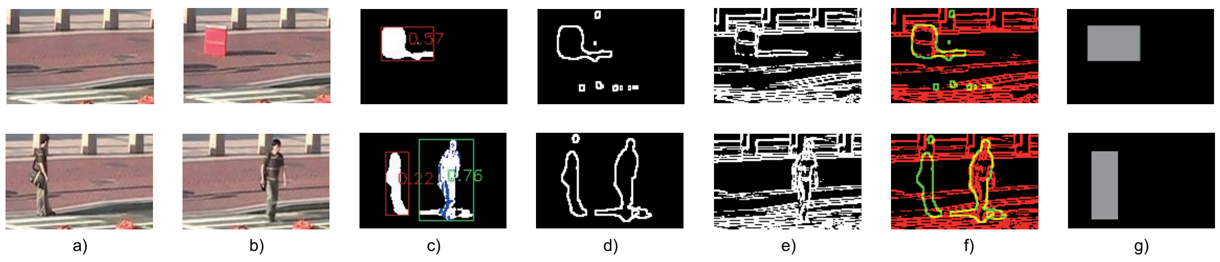

Fig. 4. Two exemplary test scenarios: upper row – stopped object, bottom row – ghost. Colums: a – frame used for background initialization, b – current frame, c – analysis results, d – foreground mask edges, e – current frame edges, f – combined edge images, g – S coefficient. Detailed description in text. Images from the `changedetection.net` database.

from the *Intermittent Object Motion* set. Comparison of the PBAS_FPGA and improved PBAS_FPGA+ version is presented in Tab. III

Quantitative analysis of the presented data shows that the proposed mechanism improves the results practically in all cases. Nevertheless, a careful visual analysis of the sequences and segmentation results reveals some problematic situations:

- on the *Parking* sequence the car is quite similar to the background (in terms of colour). In addition, the vehicle trajectory causes that the correct detection is not possible from the beginning of the movement,
- on the *Sofa* sequence the objects have colours very similar to the background. This makes the correct segmentation and edge detection very difficult,
- on the *Winter Drive* sequence there are strong shadows, which result in the presence of a large number of edges in the scene. This leads to an incorrect classification of a ghost as a real object.

Additionally, during the evaluation process of the algorithm few situations were notices, that hinder the operation of the method. Firstly, the occurrence of shadows distorts the mask and the edges of objects – i.e. PBAS is quite sensitive to shadows. On the other hand, the edges between the shadows and the background on the current frame are usually not detected as the difference is too small (the Sobel module). Therefore, the described edge comparison approach fails in such cases.

Secondly, some test sequences were recorded with automatic white balance or gain control of the camera. Therefore, the appearance of a large object (e.g. a truck) caused a change in the scene lighting and numerous, though temporary, segmentation errors.

When working on the method, it was also observed that the acceleration of image processing in the software model using either resolution reduction or frame dropping (i.e. processing each N-frame) can give a misleading picture of the effectiveness of the method or evaluated modifications.

Exemplary operation of the proposed approach is illustrated in Fig. 4. In the top row the case of stopped object is presented. At a first empty scene (column a), a red box is placed (column b). In column c the analysis results are shown: the detected object, its bounding box (red colour means no movement) and edge similarity value ($EC = 0.57$). It is worth noting that the presence of a shadow causes distortion of the object mask. In the next two columns (d and e) detected edges respectively for the foreground mask and the current frame are presented. In column f the two edge images are combined (red – foreground edges, green – mask edges, yellow – common part). It can be noticed that the edge layout is quite similar. In the last column (g) the coefficient S value is shown. The considered object is stationary, as it is quite high. Finally, due to EC value above the 0.5 threshold the object will be not included into the background model.

In the second case (bottom row) a "ghost situation" is illustrated. During background initialization a man was standing in front of a pedestrian crossing has been incorporated into the background model (column a). Then he moved away (column b). As result, two objects are detected (column c): a ghost (left, red bounding box, $EC = 0.22$) and the "real" one (right, green frame i.e. a moving object, $EC = 0.76$). Analysis of the image in column f clearly shows that the ghost edges (green) do not correspond to anything in the current frame (red). In addition the ghost is stationary for a long time – column g. Due to the low EC value and the stationarity it will be eventually incorporated into the background model

V. Hardware Implementation of the Improved PBAS Algorithm

This section provides an overview of background modelling approaches and discusses the proposed hardware module, including details of the PBAS method and one-pass connected component labelling and analysis. Also logical resources utilisation, energy consumption and computing power are summarized. Finally, the working system is presented.

A. Related Work

In the literature several articles can be found that describe hardware implementation of background generation and foreground object detection in FPGA. In the work [15] an implementation of GMM (Gaussian Mixture Models) is described. The module allowed to process a High Definition video stream with the frame rate of 20 fps. Several FPGA devices were targeted in simulation, but no final working system was presented (no external memory operations described or implemented).

In the article [16] a hardware implementation of a background generation algorithm based on Horprasert's method was presented. The targeted platform was Xilinx Spartan 3 FPGA family. The authors modified the original algorithm by adding the shadow detection mechanism which improved the

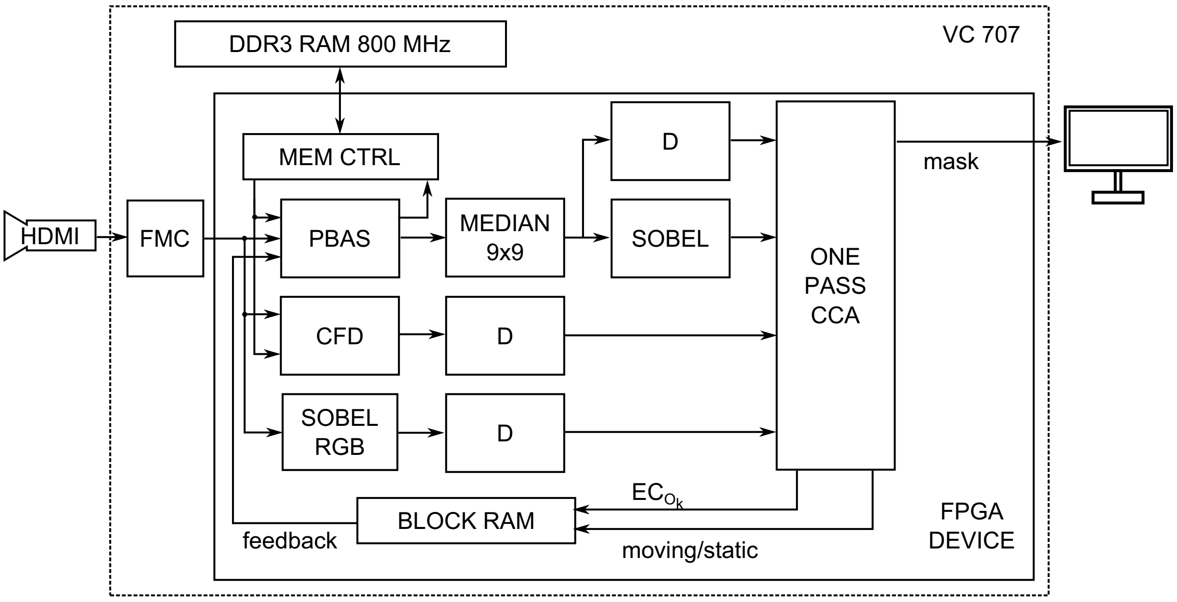

Fig. 5. Block schematic of the proposed hardware foreground object detection system.

segmentation results. The system is an example of hardware-software co-design approach. Part of the computation and control functions were performed by a Microblaze softprocesor. Moreover, two different logic description methods were used: the high level Impulse-C language (object detection) and VHDL (auxiliary modules). There was no background update mechanism in the proposed design. The initial model was created by the Microblaze softprocesor based on the first 128 frames of the video sequence. The morphological opening and closing were used for post-processing. Also a foreground object mask labelling mechanism was implemented. The system was able to process 32 frames of 1024×1024 resolution per second. The estimated power dissipation was 5.76 W.

In the work [17] a hardware implementation of Codebook method was presented. The module was tested on a Spartan 3 FPGA device. The general processing steps and methodology were similar to the one described in the paragraph above. Only the background generation algorithm was changed. Impulse-C was used to implement the most important modules. The Codebook method was altered to allow fixed point implementation. The resulting system was able to process 60 frames of 768×576 pixels images per second. The estimated power consumption was 5.76 W. In the article quantitative results of foreground object segmentation quality were presented which indicated that it was the best method among those analysed by the authors.

In paper [18] a modified sigma-delta approach implemented in Virtex 4 FPGA device was presented. It used two background models: one updated always and one updated conditionally. This unique approach allowed for better detection of foreground objects which temporarily stopped at the scene. The algorithm was a part of larger system implemented on reconfigurable platform. There were also modules responsible for edge detection, reflection detection, perspective correction, Hough transform and labelling. The system was able to process 128×128 pixels images with the frame rate of 117 fps or 2528 fps (depending on version). A very low energy consumption was pointed out as an important feature of the system.

In the work [19] a background generation method based on the Clustering algorithm and foreground object segmentation using information about intensity, colour and texture was presented. The system was working with CIE Lab colour space. The NGD (Normalized Gradient Difference) was used as a texture descriptor. A Sobel gradient computation module was used to enrich the background model with information about edges. The proposed system was tested using the Wallflower [8] dataset. The implemented hardware module was verified on the ML605 evaluation platform with Virtex 6 FPGA device. The system was able to process HD images (1920×1080) with the frame rate of 60 fps. The characteristic features were: efficient communication with external DDR3 RAM memory – the transfer reached about 5000 MB/s and compatibility with HDMI cameras. The power consumption was 10.44 W.

The visual background extractor (ViBe) method implementation on Virtex 6 FPGA device was presented in [12]. The system was able to work with maximum frequency of 140 MHz with colour video stream. The hardware realization on development board was able to process 640×480 pixel images with 50 frames per second. The main limitation was the external RAM throughput.

The quite large number of research articles in last two years proves that hardware implementation of background generation and foreground object segmentation algorithms is still an open and important research topic. Moreover the efficient access to external RAM memory is a key factor in building a real working system as well as achieving high performance video processing.

B. Overview of the System

The block scheme of the proposed hardware system is presented in Fig. 5. It corresponds to the algorithm depicted in Fig. 3, but also contains a number of auxiliary hardware modules which are necessary to implement the system on the VC707 evaluation board. Furthermore, in order to save hardware resources all arithmetic operation were realised as fixed point. A 16-bit representation was used in most cases.

The source of the video stream is a digital HDMI camera. Then, the stream is received through the Avnet DVI IO FMC module (FPGA Mezzanine Card) with HDMI input and output (FMC), which is able to convert the high frequency serial differential signal to a parallel format. The current frame is then transferred to three modules: PBAS (detailed description in Subsection V-C), CDF (consecutive frame differencing) and SOBEL RGB.

The CFD module is responsible for computing the sum of absolute difference between the current frame and the previous one, which is stored in the external RAM as a part of the background model. It its the hardware implementation of Equation (7) followed by thresholding. The SOBEL RGB module is a realisation of the Sobel edge detector. The 3×3 context generation is done using the classical delay line approach [20]. The vertical and horizontal components are combined using sum of absolute values. Finally, a thresholding is performed. The MEDIAN 9x9 is a simple binary median filter based on delay lines and a summation tree. The D modules are delays implemented as Block RAM FIFO's or flip-flops. They allow to synchronize the processing. The SOBEL module is responsible for calculating object mask edges. The ONE PASS CCA is described in details in Subsection V-D. It returns two values: the status of a bounding box (moving or static object) and the edge coefficient (EC). These parameters are stored in a Block RAM memory and read in the next iteration. In this way the feedback from the connected component analysis to the PBAS foreground segmentation module is realised.

The model, as well as previous frame are stored in an external DDR3 RAM memory. In previous works a special controller (MEM CTRL) was designed to perform a sequential read and write of a 1024-bit vector. Details are available in [19].

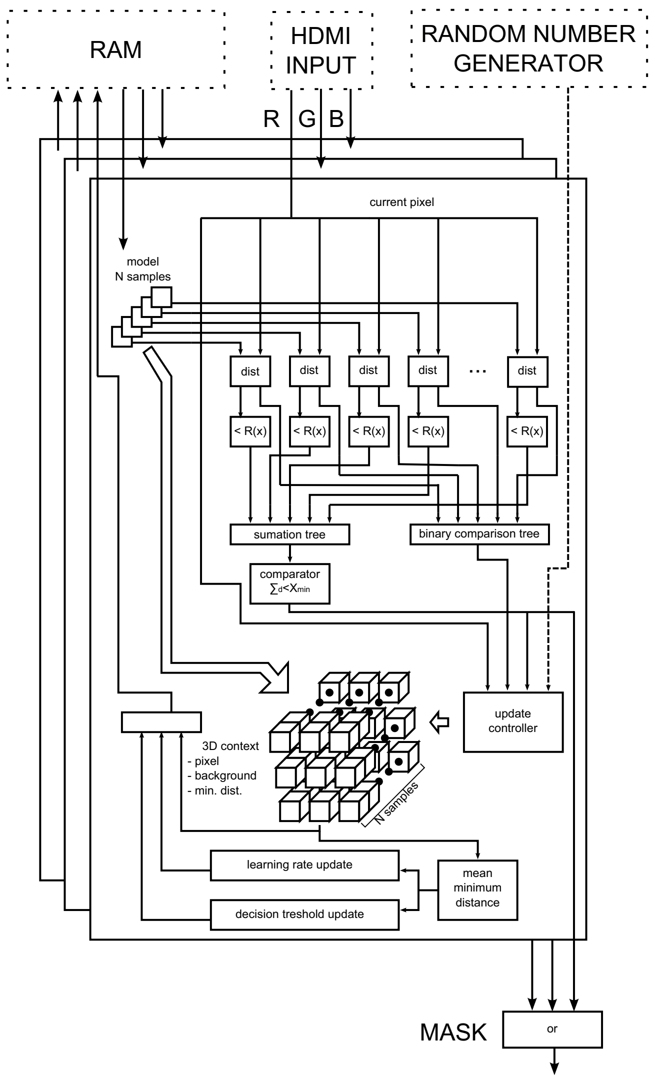

Fig. 6. Block schematic of the proposed PBAS hardware module.

C. PBAS Module

The block schematic of the PBAS module is presented in Fig. 6. Three colour channels are split and directed to three separate instances of the processing unit. Pseudo-random number generation is carried out using the concept described in [21]. It is worth noting that the authors of paper [21] made the VHDL code of different RNG versions available. This allows a quick integration of the module with the whole system.

Each single colour component processing unit loads the background model generated in previous pass of the algorithm from the external RAM memory. In the first step, the distance between the current pixel value and all previously generated samples is calculated (block `dist` – equation (3)) and compared with the threshold $R(x_i)$. This information is passed to two modules. The first one is a summation tree. The sum of distances exceeding $R(x_i)$ is compared with a threshold and the foreground/background classification is made (Equation (2)). The foreground masks obtained for each colour component RGB are combined with an OR operator. The second module is a block that computes the minimum distance value and its corresponding index using a binary comparison tree.

Although the segmentation result is calculated, the background model must be updated before the next iteration. To do this, a 3D context is created. It has the basic 3×3 sliding window layout, but each single element consists of the current pixel value ($I(x_i)$), N background model samples ($B_k(x_i)$), N minimum distances ($D_K(x_i)$), rates ($R(x_i)$,$T(x_i)$) and previous pixel value.

Four random numbers are generated by the RANDOM NUMBER GENERATOR block, which are received by the `update controller` module. The first is used to determine if an update should be performed. If so, a random sample from the central pixel, a random neighbouring model and a random sample from this model are selected and replaced with the corresponding current pixel values.

Finally, the mean minimum distance value ($\bar{d}_{min}(x_i)$) for the central pixel is computed. This requires the use of a summation tree and divider (module *mean minimum distance*). The information is then used to update the decision threshold (Equation (5)) and learning rate parameter (Equation (6)).

The background model is initialized using a module not shown in the block diagram. All parameters are set to default values and the sample buffer is filled with randomly picked pixels from a 3×3 context. The system is then switched to normal operation mode by pressing a button.

D. One Pass Connected Component Analysis

The one pass connected component analysis hardware module is presented in Fig. 7. Its main task is to label the segments of binary object mask and count various parameters within it. Four features are computed: the segment area, the number of moving pixels within each segment, the number of edge pixels of the segment and the number of edge pixels in original image which are covered by segment edge pixels.

The first block is the labelling unit. This module is gathering a 3×3 pixel context from the foreground object mask generated by the PBAS module. The block is also keeping track of previously assigned labels by using an ID counter and generating a 3×3 label context.

Based on the binary values in the mask context and previously assigned labels in the label context, a new label for the current pixel is assigned according to a few rules. If the mask value of the central element is one and if there is no previously assigned labels in the label context, a new label is assigned based on the ID counter value. If there are values in the label context it means, that the currently processed pixel belongs to a segment which has already been given an ID. If the labels in the context are the same, they are used as an ID for new pixel. If the values are not the same it means, that a collision occurred (the current pixel is a merging point for two differently labelled segments). Such situation happens, if a shape like V letter is labelled from top to down. Because of the scan order and context size (3×3), only two different labels can be present in the context. If the collision happens, one label should be marked as merged and the other is used as a new pixel label.

The information about the label number and collision event is transferred to four area computation blocks. Each block is a simple counter and a BRAM memory storing values for all possible labels. If the collision occurs, the number of counted pixels from both segments are summed up to one value and merged label is marked as invalid. To prevent the module access the invalid memory locations, all pointers (labels) are appropriately redirected.

When the whole frame is processed according to presented rules, the valid memory locations store values for separate segments of the mask image. Two other blocks compute the ratio of moving pixels to the whole segment area (S_{O_k} – Equation (8)) and the number image edge pixels to the number of segment edge pixels (EC_{O_k} – Equation (9)). Two hardware dividers are used to accomplish this task.

E. System Integration

The project was described in VHDL and Verilog hardware description languages and synthesised for a Virtex 7 (XC7VX485T) FPGA device using Xilinx ISE 14.6 Design Suite. As the target platform the Xilinx VC707 evaluation board with and Avnet DVI I/O FMC module was chosen. Simulations performed in ISim software confirmed that the hardware modules are fully compliant with software models designed in C++.

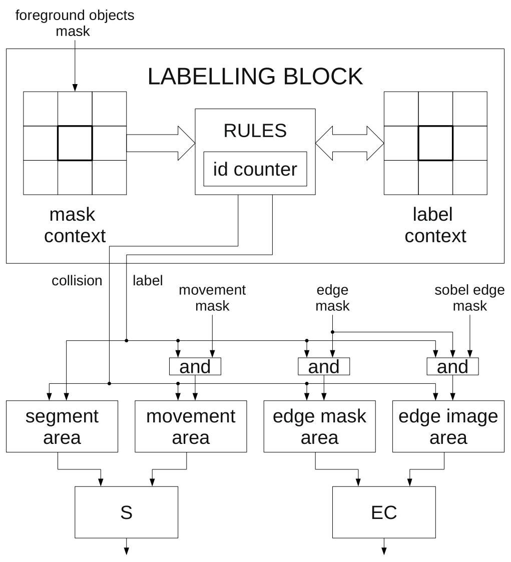

Fig. 7. Connected components analysis block.

It is worth noting the VHDL files describing the main processing module, as well as sub-modules: summation trees, binary comparison tree, context etc., are generated automatically using a script prepared in MATLAB. This allows for easy modification of parameters, especially the number of samples (N). A similar approach is used to create the median module. This makes the design very flexible and easy to implement on other hardware platforms (e.g. with lower transfer rate to external memory).

The reported maximal operating frequency after place and route phase was 101 MHz, which is more than enough for real time processing of a 720×576 colour video stream with 50 fps. The power consumption reported by Xilinx XPower Analyzer for the device (On-Chip) is about 3.483 W. The resource usage is summarized in Tab. IV.

The working system is presented in Fig. 8. It consists of an FPGA evaluation board (VC707 from Xilinx), an LCD screen and a HDMI camera (not visible). Two scenarios, analogous to the discussed is Subsection IV-A, are shown. In the upper row a box is present during background model initialization (Fig. 8a). When it is removed, a ghost appears (Fig. 8b). The detected objects are marked as semi-transparent red, and the corresponding bounding box as semi-transparent blue. After some time it is incorporated into the background model (Fig. 8c). In the bottom row, on a initially empty scene ((Fig. 8d) a box is inserted ((Fig. 8e). It is correctly detected by the system (Fig. 8f). Concluding, due to the introduced feedback from the analysis module, the foreground mask in both situations is correct.

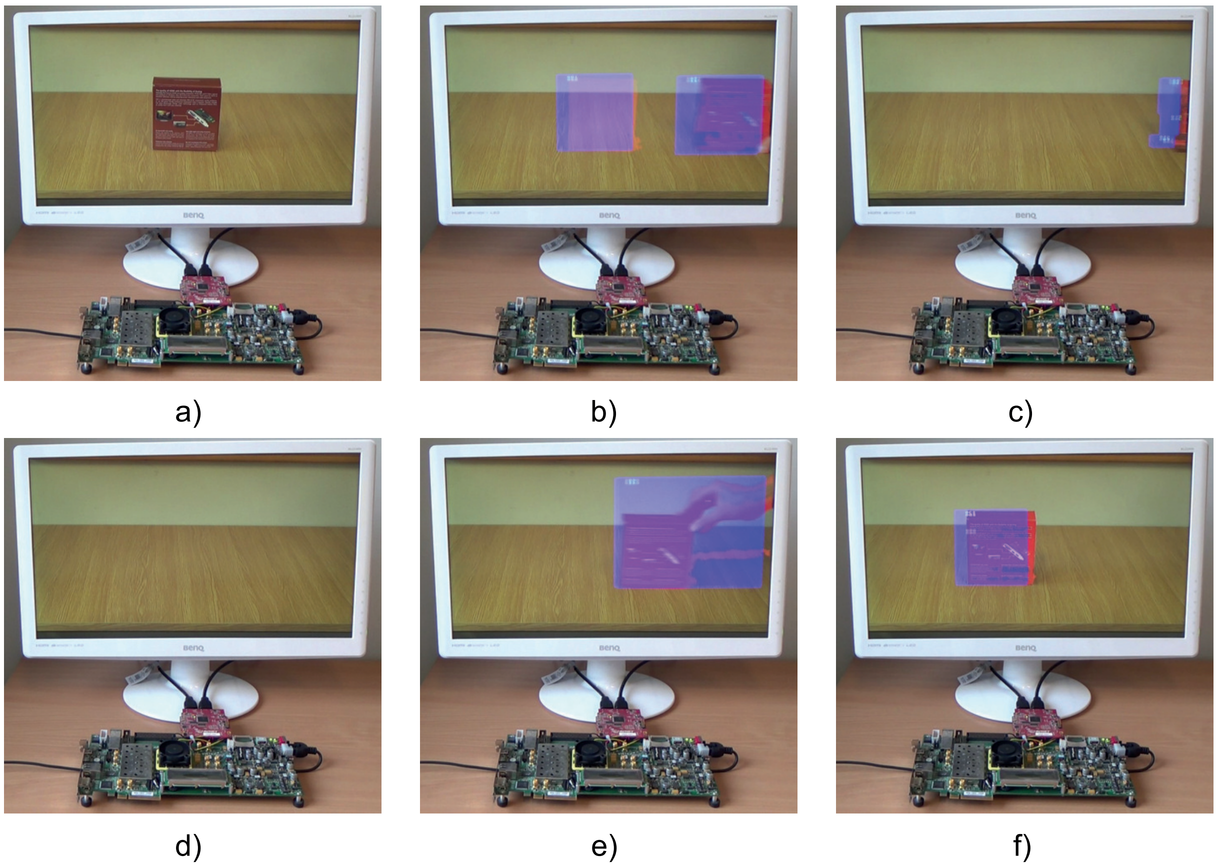

Fig. 8. Working system. Resolution 720 × 576, 50 fps, $N = 18$. Images a, b and c – ghost scenario, images d, e and f – stopped object scenario. Description in text.

TABLE IV
PROJECT RESOURCE UTILISATION

Resource	Used	Available	Percentage
FF	41969	607200	6 %
LUT 6	39780	303600	13 %
SLICE	15931	75900	20 %
DSP 48	14	2800	1 %
BRAM_18	8	2060	1 %
BRAM_36	279	1030	27 %

VI. CONCLUSION

The article demonstrates a hardware implementation of an improved method for foreground object segmentation based on the PBAS algorithm. The starting point were the conclusions drawn from research described in [6], where it turned out that the original method fails in the case of distinguishing static objects and ghosts. A solution, which is based on the foreground object properties analysis was proposed. For this purpose, the stationarity of each connected component is determined, as well as the object edges are compared with those present in the current frame. Both parameters provide feedback for the PBAS module and allow to differentiate stopped objects from ghosts. Evaluation of the results for test sequences *Intermittent Object Motion* from the `changedetection.net` dataset showed an advantage over the original proposal. The proposed algorithm was implemented in VHDL and Verilog languages and successfully verified on the VC707 hardware platform with reprogrammable FPGA device. The designed vision system allows image processing with a resolution of 720 × 576 and 50 frames per second in real time.

As part of further work on the solution, whose principal objective is to increase the reliability of foreground object segmentation, the following issues will be addressed: the use of optical flow for camera jitter compensation mechanism or in a general case allowing free camera movement (e.g. solutions with PTZ cameras), detection and elimination of shadows and the correct segmentation in case of movement in the background.

The obtained results show that, using high-end FPGAs, which are equipped with considerable logic and memory resources and have quick access to the external RAM, it is possible to create an advanced real-time video system, which is able to perform both image processing and analysis. The designed system can be used in numerous solution e.g. surveillance system, UAV and autonomous vehicles.

REFERENCES

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

[2] Y. Benezeth, P. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, “Comparative study of background subtraction algorithms,” *Journal of Electronic Imaging*, vol. 19, July 2010, Orange Labs [Rennes], MOIVRE Centre, Laboratoire PRISME – PRISME, Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen – GREYC, doi 10.1117/1.3456695.

[3] T. Bouwmans, “Subspace Learning for Background Modeling: A Survey,” *Recent Patents on Computer Science*, vol. 2, no. 3, pp. 223–234, 2009.

[4] M. Hofmann, P. Tiefenbacher, and G. Rigoll, “Background segmentation with feedback: The Pixel-Based Adaptive Segmenter,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2012, pp. 38–43, ISSN 2160-7508, doi 10.1109/CVPRW.2012.6238925.

[5] D. G. Bailey, *Design for Embedded Image Processing on FPGAs*. John Wiley and Sons, Ltd, 2011.

[6] T. Kryjak, M. Komorkiewicz, and M. Gorgon, “Hardware implementation of the PBAS foreground detection method in FPGA,” in *Proceedings of the 20th International Conference Mixed Design of Integrated Circuits and Systems (MIXDES)*, 2013, pp. 479–484.

[7] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, “Changedetection.net: A new change detection benchmark dataset,” in *IEEE Computer Society Conference onComputer Vision and Pattern Recognition Workshops (CVPRW)*, June 2012, pp. 1–8, ISSN 2160-7508, doi 10.1109/CVPRW.2012.6238919.

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

[9] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 1999, pp. 2 vol. (xxiii+637+663).

[10] OpenCV, “Website: `http://http://opencv.org/` (last access: 31.01.2014),” 2014.

[11] M. Van Droogenbroeck and O. Paquot, “Background Subtraction: Experiments and Improvements for ViBe,” in *IEEE Change Detection Workshop*, 2012, pp. 32–37.

[12] T. Kryjak and M. Gorgon, “Real-time implementation of the ViBe foreground object segmentation algorithm,” in *Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2013, pp. 591–596.

[13] R. H. Evangelio and T. Sikora, “Complementary background models for the detection of static and moving objects in crowded environments,” in *8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2011, pp. 71–76, doi 10.1109/AVSS.2011.6027297.

[14] T. Kryjak, M. Komorkiewicz, and M. Gorgon, “Reconfigurable video surveillance system for detecting intrusion into protected areas (in polish) – Rekonfigurowalny system wizyjnego nadzoru do detekcji naruszenia obszarw chronionych,” *PAK – Pomiary Automatyka Kontrola*, vol. 7, pp. 584–586, 2012.

[15] 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, DIBET, University of Napoli Federico II, Napoli, Italy, ISSN 1861-8200.

[16] R. Rodriguez-Gomez, E. J. Fernandez-Sanchez, J. Diaz, and E. Ros, “FPGA Implementation for Real-Time Background Subtraction Based on Horprasert Model,” *Sensors*, vol. 12, no. 1, pp. 585–611, 2012, ISSN 1424-8220.

[17] ——, “Codebook hardware implementation on FPGA for background subtraction,” *Journal of Real-Time Image Processing*, pp. 1–15, 2012, ISSN 1861-8200.

[18] M. Wojcikowski, R. Zaglewski, and B. Pankiewicz, “FPGA-Based Real-Time Implementation of Detection Algorithm for Automatic Traffic Surveillance Sensor Network,” *Journal of Signal Processing Systems*, vol. 68, pp. 1–18, 2012, issue 1, Gdansk University of Technology, Gdansk, Poland, ISSN 1939-8018,.

[19] 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, doi 10.1007/s11554-012-0290-5.

[20] B. Kruse, “A Parallel Picture Processing Machine,” *IEEE Transactions on Computers*, vol. C-22, 12, pp. 1075–1087, 1973.

[21] D. B. Thomas and W. Luk, “FPGA-Optimised Uniform Random Number Generators Using LUTs and Shift Registers,” in *International Conference on Field Programmable Logic and Applications (FPL)*, 2010, pp. 77–82.