J Sign Process Syst (2012) 68:1–18
DOI 10.1007/s11265-010-0569-3

FPGA-Based Real-Time Implementation of Detection Algorithm for Automatic Traffic Surveillance Sensor Network

Marek Wójcikowski · Robert Żaglewski · Bogdan Pankiewicz

Received: 15 January 2010 / Revised: 1 December 2010 / Accepted: 2 December 2010 / Published online: 15 January 2011

Abstract This paper describes the FPGA-based hardware implementation of an algorithm for an automatic traffic surveillance sensor network. The aim of the algorithm is to extract moving vehicles from real-time camera images for the evaluation of traffic parameters, such as the number of vehicles, their direction of movement and their approximate speed, using low power hardware of a sensor network node. A single, stationary, monochrome camera is used, mounted at a location high above the road. Occlusions are not detected, however simple shadow and highlight elimination is performed. The algorithm is designed for frame-rate efficiency and is specially suited for pipelined hardware implementation. The authors, apart from the careful selection of particular steps of the algorithm and the modifications towards parallel implementation, also proposed novel improvements such as backgrounds' binary mask combination or non-linear functions in highlight detection, resulting in increasing the robustness and efficiency of hardware realization. The algorithm has been implemented in FPGA and tested on real-time video streams from an outdoor camera.

Keywords Algorithms implemented in hardware · Image-processing hardware · Video analysis · Sensor networks

M. Wójcikowski (✉) · B. Pankiewicz
Gdańsk University of Technology,
Gdansk, Poland
e-mail: wujek@ue.eti.pg.gda.pl

B. Pankiewicz
e-mail: bpa@ue.eti.pg.gda.pl

R. Żaglewski
Intel Shannon Ltd,
Shannon, Ireland
e-mail: robert.c.zaglewski@intel.com

1 Introduction

Complex traffic surveillance systems are often used for controlling traffic and detecting unusual situations, such as traffic congestion or accidents. Most such systems are built using high resolution cameras connected via a high-bandwidth link to the processing center. The need for automated processing of video data is obvious and many solutions of systems for traffic analysis can be found in the literature [1–10]. This paper describes an approach which uses a single, stationary, constant zoom, monochrome camera to detect moving and recently stopped vehicles. The result of the algorithm is a binary mask image of blobs representing the detected objects and a table with the parameters of the objects. A low resolution camera can be used, since the detected objects (vehicles) are large enough and their details are not important for this application. The camera is mounted at a location high above the road, e.g. on a street-lamp pole, to reduce occlusions of vehicles and provide a large field of view. The camera observes dynamic events in a scene with a fixed or slowly changing background, locally occluded by moving vehicles. It is assumed that no a priori knowledge about the scene is needed; possible camera vibrations have to be reduced by a separate block. The proposed algorithm supports day and night operation, where the scene might be illuminated by an additional light source (e.g. street lamps) or an infra-red camera could be used. The algorithm runs at video-rate performance and enables low power realization in a sensor network node, which can be powered from a solar cell. Most of the design decisions have been made taking into the account the possibility of the implementation in the sensor network node with limited hardware and power resources. The low power operation is achieved due to the low resolution of the processed image, which enables to use

a low frequency clock. This algorithm has been developed for and tested in an autonomous low-cost sensor network node for the car traffic flow evaluation. The set of such nodes enabled to estimate traffic in a large area of a city. Some early results of the authors' work have been presented in [6], in this paper the final algorithm has been described in details, the Hough transform block has been added, the final processing block has been revised and improved and the edge detection blocks have been introduced.

The layout of this paper is as follows: the overview of the most important developments in the image segmentation area is presented in Section 2. In Section 3 the authors present a low-level image-processing algorithm for the detection of moving objects. Section 4 describes the transformation of the blobs obtained from the image-processing algorithm into a table containing the coordinates of the detected moving objects with their basic shape parameters. The results of hardware implementation and conclusions are presented in Sections 5 and 6, respectively.

2 Related Work

Moving object detection and segmentation techniques have been investigated for many years. Two main approaches to the problem of recognizing the vehicles on the video image can be distinguished:

1) A model-based approach using feature-extraction methods, where recognition is achieved by matching 2-D features extracted from the image with the features of a 3-D model [1, 2, 11–15]
2) A non-model-based approach, where the following three major methods of moving object segmentation, can be distinguished: optical flow [16–18], frame differencing and background subtraction [4, 19–25].

Background subtraction is probably most often used for moving object segmentation. The idea of the background subtraction can be described with the following inequality:

$$|\mathbf{I}_t - \mathbf{B}_t| < \theta \tag{1}$$

where $\mathbf{I}_t$ is the matrix of the pixel intensities of the current frame, $\mathbf{B}_t$ denotes the actual background image for time t and θ is a constant threshold.

The main effort in the background subtraction method is concentrated on maintaining the correct background image. The running average [3] enables the quick calculation of an approximated value of the background (2):

$$\mathbf{B}_t = \alpha \cdot \mathbf{I}_{t-1} + (1-\alpha) \cdot \mathbf{B}_{t-1} \tag{2}$$

where α is a learning ratio.

For image processing, a median operation usually gives better results than the average, thus the running median has been introduced in [27] and [4], where the running estimate of the median is incremented by one if the input pixel's intensity is larger than the estimate and decreased by one if it is smaller. The estimate converges to the median value, since half of the input pixels are larger than and half are smaller than the estimated value. Calculating the median estimate in this way can also be very efficiently realized in hardware.

Methods using Gaussian distribution or a running Gaussian average [28–30] provide a solution to the threshold selection problem. Each pixel is modeled as an independent statistical process of Gauss distribution and non-background pixels are found using the following inequality:

$$|\mathbf{I}_t - \boldsymbol{\mu}_t| < k \cdot \boldsymbol{\sigma}_t \tag{3}$$

where $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$ are mean and standard deviation matrices of Gaussian distribution for image pixel intensities and the constant k typically has a value between 2 and 3.

Updating the background image with running Gaussian is calculated as shown in the following equations:

$$\mu_t(x,y) = \alpha \cdot I_{t-1}(x,y) + (1-\alpha) \cdot \mu_{t-1}(x,y) \tag{4}$$

$$\sigma_t^2(x,y) = \alpha[I_{t-1}(x,y) - \mu_{t-1}(x,y)]^2 + (1-\alpha) \cdot \sigma_{t-1}^2(x,y) \tag{5}$$

In further considerations, the pixel's coordinates (x, y) will be omitted to aid the readability of this paper, unless necessary.

Simplification of the Gaussian models can be found in [31], where the absolute maximum, minimum and the largest consecutive difference values for every pixel are used. Adding color information can improve the sensitivity of the moving object detection. For this purpose, several color models can be used, such as RGB or HSV [29, 32].

Modeling each pixel as an independent statistical process with Gaussian distribution does not enable the overcoming of the problem of non-stationary background (waving trees or water waves), where the background pixels might have several distributions. This problem can be solved by using the Mixture of Gaussians (MOG) [5, 33–37], where each pixel is modeled by a few distributions which are constantly updated. In [38] the authors, apart from MOG, use a statistical model of gradients with a hierarchical approach of image analysis with pixel-level, region-level and frame-level processing. An interesting FPGA implementation (using off-chip RAM) of the modified MOG algorithm is presented in [39], where also the blob labeling is implemented in hardware.

There are also other methods found in the literature, such as kernel density estimators based on a simple adaptive filter introduced in [40], a method using Kalman filter [34], a linear prediction (Wiener filter) with autoregressive process of order 30 [41], mean-shift based estimation [42] or eigenbackgrounds.

Background subtraction enables the detection of moving and stopped objects. Depending on the background model update technique, the stopped objects can be detected for a certain amount of time, until they become a part of the background. The disadvantage of this approach is the effect of detecting the places where the stopped object, which was a part of the background, started to move. Such a relocation of the background object is called a ghost. The empty place where the object was before it started to move is detected until it becomes a part of the background. The other issue is how to update the background. Simple approaches use a non-selective background update—information from every pixel of the current image is included in the background model, despite the result of the segmentation. In this way, the pixels belonging to the moving objects are also included in the background, decreasing the selectivity of the segmentation. In [3], a selective background update has been introduced, where only pixels that are not recognized as moving objects are allowed to be included in the background model. Selectivity improves the quality of the background model, but it also creates a risk of the occurrence of the dead-lock phenomenon, which appears when some part of the background changes and the pixels that are falsely detected as being part of the moving object will never be included in the background and are always indicated in the segmented picture. To reduce this problem, two backgrounds can be used [7, 40, 43, 44], as a combination of both selective and non-selective approach.

Owing to the algorithm imperfections, some pixels of the original image being a part of the moving vehicle are not indicated in the binary mask. Such pixels are called false negatives (FN). When the pixels of the original image that are part of the stationary background are recognized as part of the moving objects, they are called false positives (FP). TP denotes the number of true positive pixels, i.e. pixels that are a part of the moving object and are correctly identified. The following detection quality measures can be defined, i.e. the fill ratio *FIL* and the precision ratio *PR*, similarly as in [46]:

$$FIL = \frac{TP}{TP + FN} \cdot 100\% \tag{6}$$

$$PR = \frac{TP}{TP + FP} \cdot 100\% \tag{7}$$

One of the reasons for errors in image segmentation is the existence of shadows of various types: shadows cast by objects onto the background, shadows cast by objects onto themselves and shadows cast by other objects. The most important are the shadows which are cast by objects onto the background and which move along with them. In the night, highlights can be observed, such as the reflections of car lights from background surfaces. Even a snow can cause additional problems [47]. Some authors divide the image into squares and manipulate the mean values of the pixels' intensities to eliminate shadow and to preserve the texture [48], or use color, spatial and temporal information with an a posteriori probabilistic estimator to determine shadows [49]. In [50], a set of 17 image region types is used and heuristic rules to classify pixels as shadow are applied. Many solutions characterize shadow by the same color but lower hue or brightness [8, 26, 53]. The detailed review of shadow elimination techniques can be found in [8] and [54]. Elimination of shadows and highlights is very important in a non-model-based approach, since they could cause object merging or shape distortions.

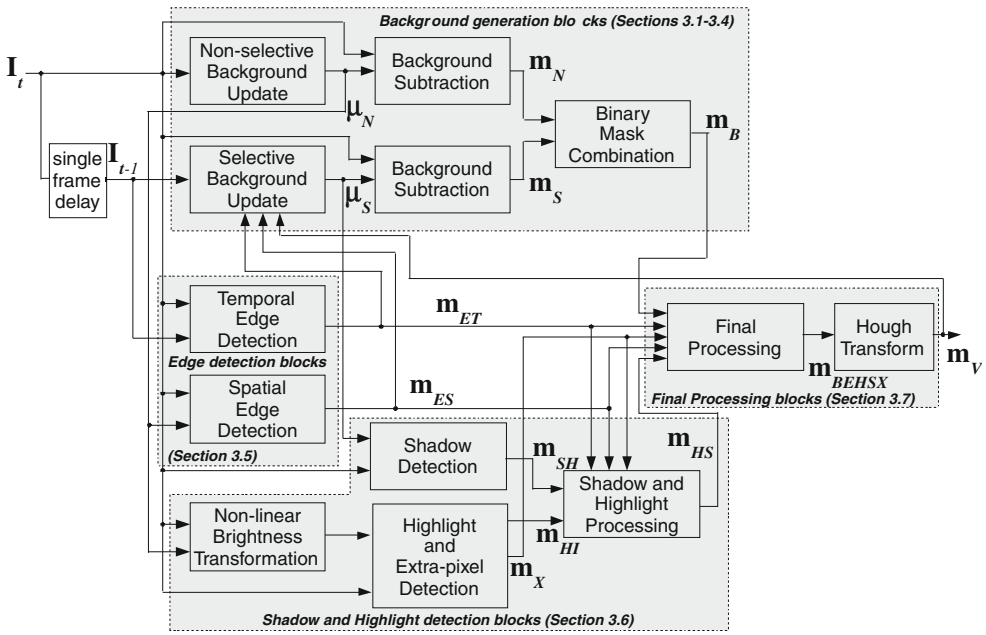

Figure 1 General diagram depicting the idea of the algorithm for the FPGA implementation.

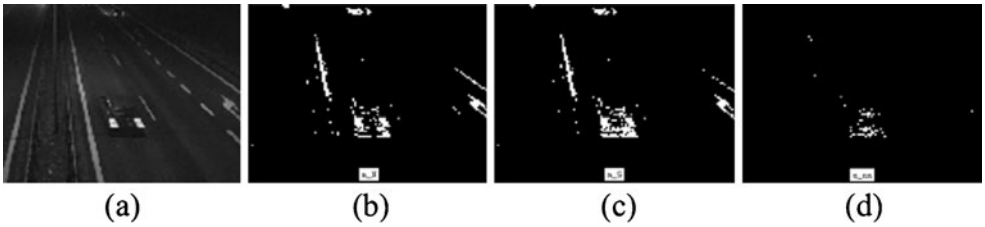

Figure 2 An example illustrating a better sensitivity of the non-selective background model, (**a**)—input image $\mathbf{I}_t$, (**b**)—mask $\mathbf{m}_N$ from non-selective background, (**c**)—mask $\mathbf{m}_S$ from selective background, (**d**)—pixels detected in the mask $\mathbf{m}_S$ and not detected in the mask $\mathbf{m}_N$.

The binary mask image obtained from the segmentation algorithm should be post-processed to delete single erroneously classified pixels with morphological operations or other methods using additional information obtained from object-level and frame-level processing [41].

The hardware implementation of image processing algorithms is becoming more popular with the constant development of more sophisticated FPGAs. Examples of implementation of image processing algorithms can be found in [51] and [52].

3 Image Segmentation Algorithm

In this paper, a non-model-based approach is presented, which transforms the camera image into a binary mask containing moving blobs. The aim of the authors is to develop a pipelined, iteration-less algorithm which can be implemented in hardware, performing simple segmentation of traffic objects with a monochrome camera mounted above the road. The use of a monochrome camera decreases the segmentation sensitivity and it also excludes the use of color information for shadow and highlight detection, but it reduces the complexity of the hardware. The authors developed the algorithm that can be implemented in pipelined fashion in the hardware, without iterations. The contribution of this paper is also a novel method of binary mask combination from two background subtraction results and the use of the non-linear functions for the detection of the highlights.

The general diagram depicting the idea of the algorithm is presented in Fig. 1, where the block structure of the system and the data-flow are shown. Each block will be described in detail in the next sections. Since the background subtraction technique is used, the image stabiliza-

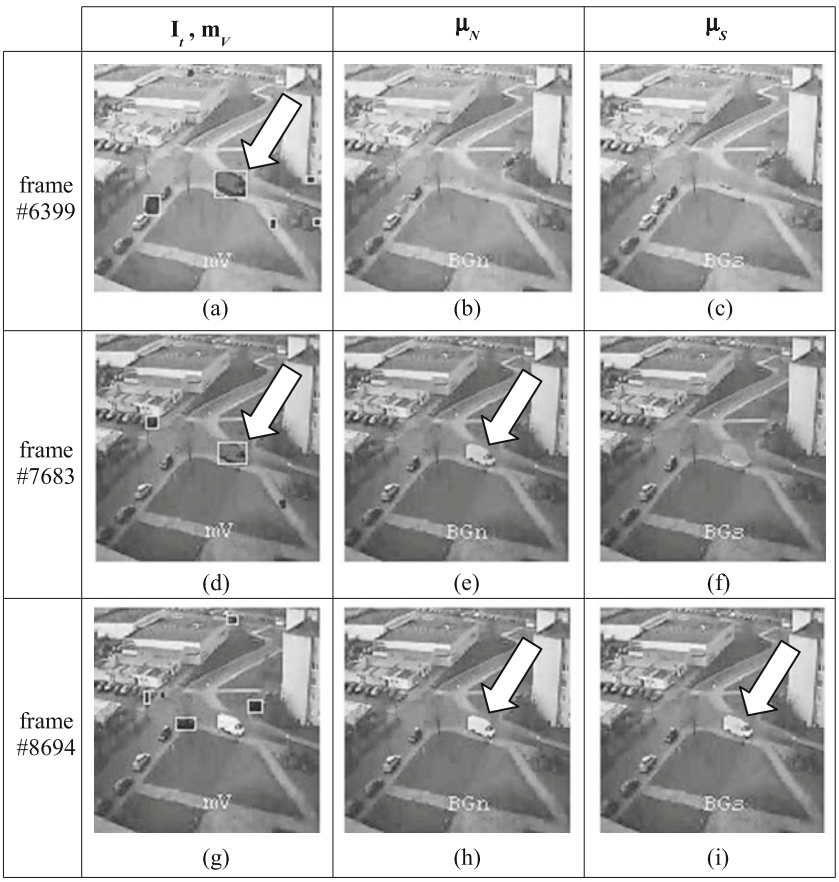

Figure 3 Detection of the recently stopped object by selective and non-selective background update blocks: (**a**)—the input image $\mathbf{I}_t$ with added mask $\mathbf{m}_V$ (darker areas indicated by the white rectangles), the recently stopped car in the center of the image is being detected by the algorithm, (**b**)—the non-selective background μ_N, the new car has not been yet added to the non-selective background, (**c**)—the selective background μ_S, the new car has not yet been included into the selective background, (**d**)—the car is still being detected, (**e**)—the car is slowly being added into the non-selective background, (**f**)—the car is not included into the selective background, because it has been blocked by the mask $\mathbf{m}_V$, (**g**)—the car is not detected any more, (**h**)—the new car is fully included into the non-selective background, (**i**)—the selective background is quickly updating, because mask $\mathbf{m}_V$ is not blocking the update of the car.

Table 1 Binary mask combination improving the final $\mathbf{m}_B$ mask quality.

Situation	Input masks		Result		
	$\mathbf{m}_S$	$\mathbf{m}_N$	Simple AND	Simple OR	Binary Mask Combination $\mathbf{m}_B$
Missing single pixel (FN) in $\mathbf{m}_S$					
FP single pixel (noise) in $\mathbf{m}_S$					

tion circuit at the input might be needed, which is outside the scope of this paper.

3.1 Models for Selective and Non-selective Background

The presented algorithm is based on the background subtraction technique and uses two background models: long-term with non-selective update and short-term with selective background update [6, 7, 40, 43]. The models for both selective and non-selective backgrounds are similar; the difference is only in updating the background with data from the current image. For the simplicity of the realization in hardware, the models assume single Gaussian distribution of pixels' intensities. The pixel is classified as foreground using (3) and the results are stored as $\mathbf{m}_S$ and $\mathbf{m}_N$ masks for selective and non-selective background, respectively.

Depending on the auto exposure system implemented in the camera, sudden changes in the illumination of the scene can cause the background subtraction-based algorithms to detect all the regions where the brightness has changed. Such a situation can be observed at night, for example near periodically flashing city neon lights. In this situation an additional average brightness control block adjusting the average brightness of the background models and the previous frame to the current image might be needed, which is outside the scope of this paper.

3.2 Non-selective Background Update Block

The non-selective background update block, along with the selective background update block, performs the main task of detecting the moving and recently stopped objects. To enable easy implementation in hardware, the running mode [7] as a background update function was chosen:

$$\mu_{N,t}(x,y) = \begin{cases} \mu_{N,t-1}(x,y) + \delta_{NI} & \text{if} \quad I_t(x,y) > \mu_{N,t-1}(x,y) \\ \mu_{N,t-1}(x,y) - \delta_{NI} & \text{if} \quad I_t(x,y) < \mu_{N,t-1}(x,y) \\ \mu_{N,t-1}(x,y) & \text{otherwise} \end{cases} \tag{8}$$

where:

$I_t(x, y)$ the brightness of a pixel situated at coordinates (x, y) of input monochrome image at the time t

$\mu_{N,t}(x, y)$ the brightness of a pixel situated at coordinates (x, y) of background image, updated non-selectively

$\delta_{NI} = 2^{-5} = 0.03125$ is a small constant evaluated experimentally. It is assumed that the brightness of the input image is in the range: $I_t(x,y) \in \langle 0, 255 \rangle$.

As can be seen from (8), the calculation of the background requires only a few simple operations. The running mode is also used for updating standard deviation σ_t. Experimental results show that this approach works correctly and also enables fast and easy implementation in hardware. The updating of $\sigma_{N,\,t}$, which is σ_t from (3) for non-selective model, is presented in (9):

$$\sigma_{N,t} = \begin{cases} \sigma_{N,t-1} + \delta_{N2} & \text{if} \quad \left|I_t - \mu_{N,t-1}\right| > \sigma_{N,t-1} \\ \sigma_{N,t-1} - \delta_{N2} & \text{if} \quad \left|I_t - \mu_{N,t-1}\right| < \sigma_{N,t-1} \\ \sigma_{N,t-1} & \text{otherwise} \end{cases} \tag{9}$$

where δ_{N2} is also a small constant of experimentally evaluated value of 0.00390625 (i.e. 2^{-8}).

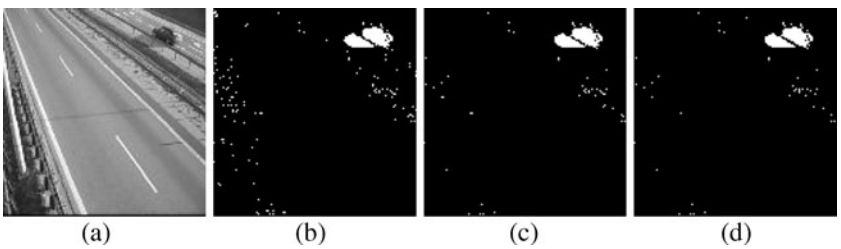

Figure 4 An example of Mask Combination Block operation: (**a**)—input picture, (**b**)—mask $\mathbf{m}_S$ from selective background, (**c**)—mask $\mathbf{m}_N$ from non-selective background, (**d**)—mask $\mathbf{m}_B$. The differences in FP pixels between (**b**) and (**c**) are caused by the differences in the background models (different update rates of selective and non-selective background, specified by the constants: δ_{N1}, δ_{N2}, δ_{S1} and δ_{S2}).

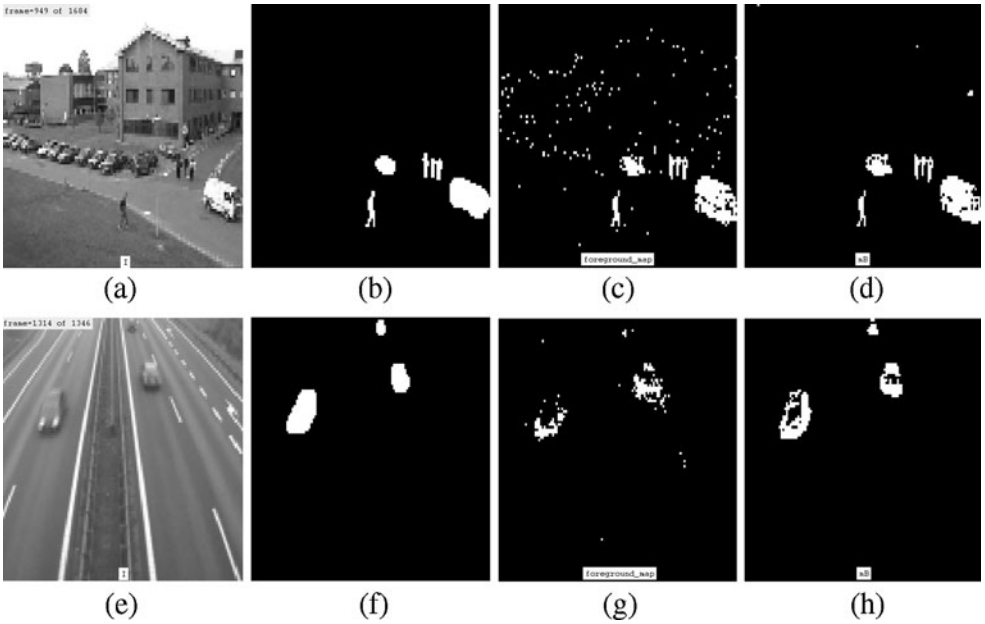

Figure 5 Comparison of the proposed background subtraction algorithm with the standard algorithm [35]: (**a**)—input image (frame #949) from PETS2001 Camera 1 sequence [45] resized to 128×128 pixels, (**b**)—manually marked ground truth, (**c**)—unfiltered result from MOG algorithm [35] with K=3 distributions, (**d**)—result from the proposed background subtraction algorithm, (**e**)—input image (frame #1314) from *obw2_d3* sequence, (**f**)—manually marked ground truth, (**g**)—unfiltered result from MOG algorithm [35] with K=3 distributions, (**h**)—result from the proposed background subtraction algorithm.

3.3 Selective Background Update Block

The selective update block works similarly to the non-selective, but uses information from the final steps of the algorithm, as can be seen in (10) and (11):

$$\mu_{S,t}(x,y) = \begin{cases} \mu_{S,t-1}(x,y) + \delta_{S1} & \text{if} \quad I_{t-1}(x,y) > \mu_{S,t-1}(x,y) \quad \text{and} \quad m_{VTS,t-1}(x,y) = 0 \\ \mu_{S,T-1}(x,y) - \delta_{S1} & \text{if} \quad I_{t-1}(x,y) < \mu_{S,t-1}(x,y) \quad \text{and} \quad m_{VTS,t-1}(x,y) = 0 \\ \mu_{S,t-1}(x,y) & \text{otherwise} \end{cases} \tag{10}$$

$$\sigma_{S,t}(x,y) = \begin{cases} \sigma_{S,t-1}(x,y) + \delta_{S2} & \text{if} \quad \left|I_{t-1}(x,y) - \mu_{S,t-1}(x,y)\right| > \sigma_{S,t-1}(x,y) \quad \text{and} \quad m_{VTS,t-1}(x,y) = 0 \\ \sigma_{S,t-1}(x,y) - \delta_{S2} & \text{if} \quad \left|I_{t-1}(x,y) - \mu_{S,t-1}(x,y)\right| < \sigma_{S,t-1}(x,y) \quad \text{and} \quad m_{VTS,t-1}(x,y) = 0 \\ \sigma_{S,t-1}(x,y) & \text{otherwise} \end{cases} \tag{11}$$

where:

$$m_{VTS,t}(x,y) = m_{V,t}(x,y) \vee m_{ET,t}(x,y) \vee m_{ES,t}(x,y)$$

$\mu_{S,t}(x, y)$ the brightness of a pixel at coordinates (x, y) of background image updated

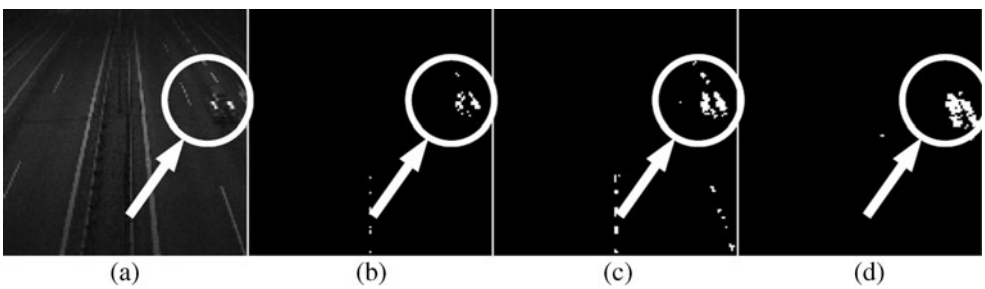

Figure 6 Additional pixels found by the edge detection in the dark scene, the moving car is marked with the circle: (**a**)—original picture (highway in the night), (**b**)—result of background detection m_B, (**c**)—result of spatial edge detection m_{ES} for θ_{ES}=20, (**d**)—result of temporal edge detection $\mathbf{m}_{ET}$ for θ_{ET}=20.

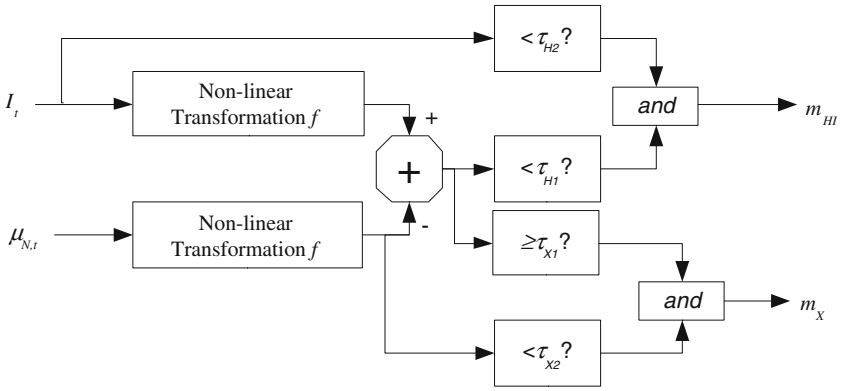

Figure 7 Flow diagram of highlight detection block.

using selectivity

$m_V(x, y)$ the element of the detected vehicle mask image of value equal to 0 or 1, where 1 denotes the detected moving objects

$m_{ET}(x, y)$ and $m_{ES}(x, y)$ the elements of value {0, 1}, obtained from temporal and spatial edge detection block, respectively.

The values of constants δ_{S1} and δ_{S2} were established experimentally: $\delta_{S1}=0.25$, $\delta_{S2}=0.03125$ for $I_{t-1}(x,y) \in \langle 0, 255 \rangle$. The input image $\mathbf{I}_{t-1}$ is used instead of $\mathbf{I}_t$ to provide the coherence with the masks $\mathbf{m}_{V,t-1}$, $\mathbf{m}_{ES,t-1}$ and $\mathbf{m}_{ET,t-1}$. The sizes of matrices $\boldsymbol{\mu}_S$, $\boldsymbol{\mu}_N$, $\mathbf{m}_V$, $\mathbf{m}_{ET}$ and $\mathbf{m}_{ES}$ are equal to the size of $\mathbf{I}_t$.

The use of two background update blocks has a very important advantage—the fast adapting selective background update block gives a better sensitivity, while the non-selective background helps to avoid the dead-lock phenomenon. An example of the frame, where the additional pixels are detected by the selective background is presented in Fig. 2.

The recently stopped objects can be detected for some time which is often required in car detection (i.e. detecting a traffic jam). The non-selective background model has longer adaptation times than the selective one, i.e. $\delta_{N1} < \delta_{S1}$ and $\delta_{N2} < \delta_{S2}$, so the recently stopped moving objects are not added to the quickly adapting selective model, because the update is blocked by the mask $\mathbf{m}_V \vee \mathbf{m}_{ES} \vee \mathbf{m}_{ET}$. After some time, the stopped objects become a part of the non-selective background and then they are quickly included in the selective background, since the mask $\mathbf{m}_V \vee \mathbf{m}_{ES} \vee \mathbf{m}_{ET}$ stops blocking of the update. This process is illustrated in Fig. 3. Using constant adaptation times (δ_{N1}, δ_{S1}, δ_{N2}, δ_{S2}) benefits in a simpler hardware, but rapid changes of the scene caused by sudden weather changes cause temporary problems in the detection. It has been observed, that typically after a few seconds, the backgrounds adapt to the new light conditions. This situation can be detected at the final stage of object segmentation, as the total area of the detected objects is comparable to the area of the whole image.

3.4 Binary Mask Combination Block

Detection results from both models have to be combined into a single binary mask $\mathbf{m}_B$. With a simple *and* operation, all the pixels that were not detected simultaneously by both models would be lost. Owing to this, a special combination of *and* and *or* operations can be used to improve the detection. In this paper, the authors refined the idea described in [43]. When in the proximity of the inspected pixel there is at least one pixel detected by both models, the *or* operation is used, otherwise the *and* operation is used, as shown in (12).

$$m_B(x,y) = \begin{cases} m_S(x,y) \vee m_N(x,y) & \text{if} \quad \begin{array}{c}(m_S(x-1,y) \wedge m_N(x-1,y)) \vee \\ (m_S(x-1,y-1) \wedge m_N(x-1,y-1)) \vee \\ (m_S(x,y-1) \wedge m_N(x,y-1)) \vee \\ m_S(x+1,y-1) \wedge m_N(x+1,y-1)\end{array} \\ m_S(x,y) \wedge m_N(x,y) & \text{otherwise} \end{cases} \tag{12}$$

The operation of the Binary Mask Combination Block is presented in Table 1, where single FN and FP pixels are considered. In those situations the results are better than simple binary operations (AND, OR). As can be seen in Fig. 4, the noise observed in the mask $\mathbf{m}_S$ (Fig. 4b) does not appear in the resulting image mask $\mathbf{m}_B$ (Fig. 4d). It must be noted that for the simplicity of the hardware, apart from the current pixel, only the four previously analyzed neighboring pixels are used in (12).

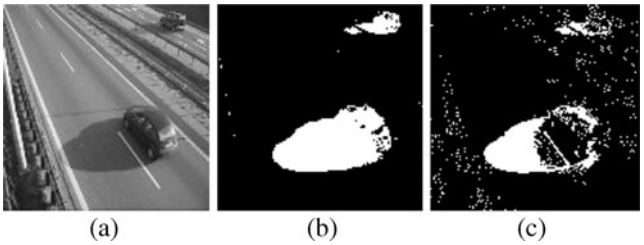

Figure 8 Simulation results of the shadow detection: (**a**)—original picture, (**b**)—result of the background detection m_B, (**c**)—the detected shadow mask m_{SH}.

The proposed background subtraction has been compared with the Stauffer's and Grimsons's MOG method [35] with K=3 distributions. For the comparison, the test sequences have been used: publicly available dataset PETS2001 [45] and the sequence taken from the bridge above the highway. As can be seen in Fig. 5, the obtained results are comparable or even better than the standard background subtraction method using the MOG.

3.5 Temporal and Spatial Edge Detection Blocks

A pure background subtraction does not detect many TP pixels, especially in dark scenes. In the worst case, the major part of the moving vehicle may not be detected in the night, except for the car lights. To overcome such a problem, an additional detection scheme has been introduced using the edge detection. The edge detection improves the segmentation quality by increasing the number of TP pixels. Two edge detection blocks have been used: temporal edge and spatial edge detection blocks. The temporal edge detection block detects the edges in the image obtained as the difference between the current and the previous frame:

$$\Delta \mathbf{I}_T = |\mathbf{I}_t - \mathbf{I}_{t-1}| \tag{13}$$

The spatial edge detection block uses the difference between the current image and the background:

$$\Delta \mathbf{I}_S = |\mathbf{I}_t - \boldsymbol{\mu}_{N,t}| \tag{14}$$

To avoid locking up the background update by continuously detected edges, the non-selective background is used in (14). Temporal edge mask image $\mathbf{m}_{ET}$ is described with (15):

$$m_{ET}(x,y) = \begin{cases} 1 & \text{if } |\Delta I_T(x,y) - \Delta I_T(x-1,y)| > \theta_{ET} \vee \\ & \quad |\Delta I_T(x,y) - \Delta I_T(x,y-1)| > \theta_{ET} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

A similar equation as (15) can be written for $\mathbf{m}_{ES}$:

$$m_{ES}(x,y) = \begin{cases} 1 & \text{if } |\Delta I_S(x,y) - \Delta I_S(x-1,y)| > \theta_{ES} \vee \\ & \quad |\Delta I_S(x,y) - \Delta I_S(x,y-1)| > \theta_{ES} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

where θ_{ET} and θ_{ES} are constant thresholds evaluated experimentally. The example of the edge detection is shown in Fig. 6. The background detection (mask $\mathbf{m}_B$) has problems in finding a dark car in the night, but the edge detections add more pixels improving the overall result. Some new FP pixels are also introduced (the lower part of Fig. 6c), but they can be easily filtered out during one of the next processing steps.

3.6 Shadow and Highlight Detection Blocks

The basic detection of shadows in monochrome images can be done simply by comparing the decrease in brightness [26]:

$$m_{SH}(x,y) = \begin{cases} 1 & \text{if } \quad \alpha \le \frac{I_t(x,y)}{\mu_{N,t}(x,y)} \le \beta \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where α and β are constant coefficients: α=0.55, β=0.95, both evaluated experimentally.

During the night, the illumination of the scene changes drastically. The light reflections from car lights are imposing the detection of many FP pixels. Detection of the highlights working similarly to that in shadow detection would cause many errors during the day. To solve this problem, the authors propose non-linear brightness transformations f, providing different behavior of the highlight detection block in the day and night. The idea of this method is presented in Fig. 7.

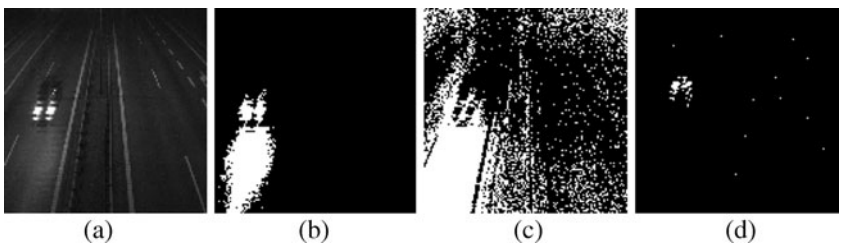

Figure 9 Simulation results of the detection of the highlights: (**a**)—original picture, (**b**)—result of the background detection m_B, (**c**)—the detected highlights in mask m_{HI}, (**d**)—mask m_X.

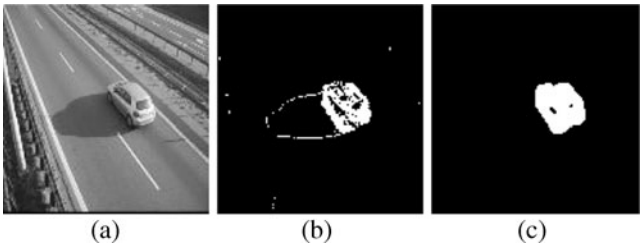

Figure 10 Simulation results showing the effect of the Hough transform: (**a**)—original picture, (**b**)—mask m_{BEHSX} (**c**)—final mask m_V for $\Theta_H = 180$.

The input image and the background image are first transformed with a non-linear function, which transforms dark pixels into bright ones and vice versa. For example, a hyperbolic function from (18) can be used:

$$f(I(x,y)) = \frac{2047}{I(x,y)+1} \tag{18}$$

where $I(x, y)$ represents the brightness of the pixel at (x,y), $I(x,y) \in \langle 0, 255 \rangle$.

In the night, when the background pixels are mainly dark and are very sensitive to any highlights, after the transformation the difference between the highlight (small value after transformation) and the background (large value after transformation) is large and the highlights can easily be detected and stored as mask $\mathbf{m}_{HI}$. During the day, the difference between the transformed background (low value) and a bright object (also low value after transformation) is smaller than the constant threshold τ_{H1}. An additional threshold τ_{H2} was introduced to exclude very bright pixels from being classified as highlights during the day. Further improvement in the number of TP pixels can be achieved by detecting very dark pixels on a bright background, also using non-linear transformations (mask $\mathbf{m}_X$ calculated

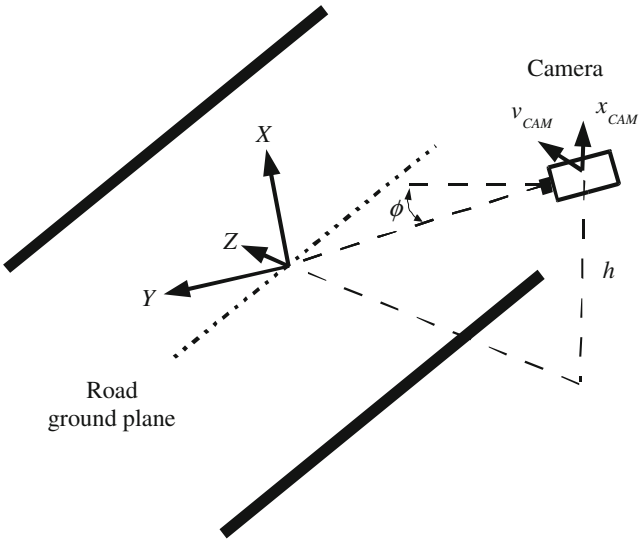

Figure 11 Geometrical model of camera and road [55].

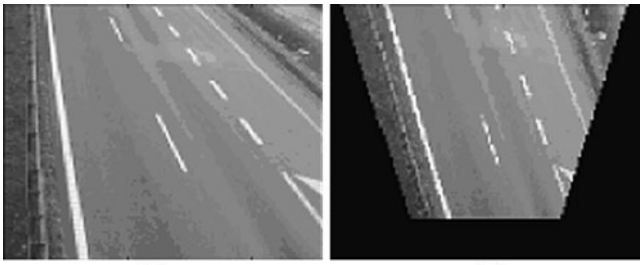

Figure 12 Example of original image (**a**) and result of transformation (**b**) for φ=43°, h=15 m, f=2.8 mm using the reverse of (22) and (23).

according to Fig. 7). The values of τ_{H1}, τ_{H2}, τ_{X1}, τ_{X2} have to be determined experimentally, the authors used the following values: τ_{H1}=−8, τ_{H2}=120, τ_{X1}=25, τ_{X2}=70. The results of shadow and highlight detection are shown in Figs. 8 and 9. As can be seen in Fig. 8, the use of the simple shadow detection technique is not perfect, but it detects the major part of the shadow and seems to be sufficient for this application. More reliable shadow detection techniques, thus more complex, are widely present in the literature, e.g. [8]. The detected highlights from the car lights are shown in Fig. 9c as the mask $\mathbf{m}_{HI}$. Finally, the mask $\mathbf{m}_X$ (Fig. 9d) identifies few more pixels of the moving object.

The shadow and highlight detections work constantly during the day and night, resulting in adding some noise at night, which is canceled at morphological operation at the final processing stage. The highlight detection depends on the brightness of the pixels, thus its operation is limited during the day, when the pixels are brighter.

3.7 Final Processing

The masks obtained in the previous steps of the algorithm are combined into a single mask $\mathbf{m}_{BEHSX}$ in accordance with (19) and (20):

$$\mathbf{m}_{HS} = dil(ero(\neg((\mathbf{m}_{ET} \wedge \mathbf{m}_{ES}) \vee \mathbf{m}_X) \wedge (\mathbf{m}_{HI} \vee \mathbf{m}_{SH}))) \tag{19}$$

$$\mathbf{m}_{BEHSX} = ero(dil((\mathbf{m}_B \wedge \neg \mathbf{m}_{HS}) \vee ((\mathbf{m}_{ET} \wedge \mathbf{m}_{ES}) \vee \mathbf{m}_X))) \tag{20}$$

where *dil()* and *ero()* denote 2×2 morphological dilation and erosion operation, respectively.

The blobs representing moving objects in the mask $\mathbf{m}_{BEHSX}$ usually contain holes (FN pixels) and many FP pixels (Fig. 10b). To improve the shape of the blobs, the authors propose to apply a generalized Hough transform with a rectangular structuring element. The size of the structuring element should correspond to the size of the

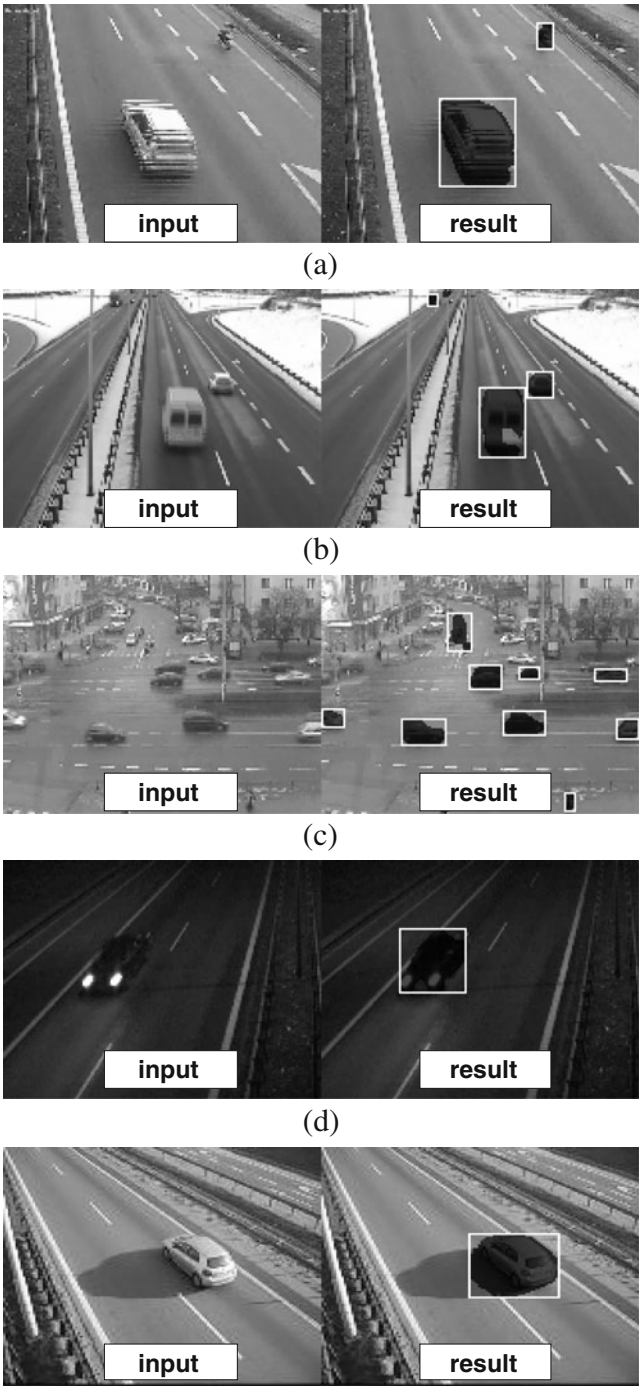

Figure 13 Simulation results of the algorithm (**a**)—frame #638 from "*obwodnica_1*" movie, (**b**)—frame #85 from "*obwodnica_2*" movie, (**c**)—frame #1879 from "*wrzeszcz*" movie, (**d**)—frame #176 from "*obwodnica_noc*" movie, (**e**)—frame #718 from "*obwodnica_6*" movie. Frames "result" on the right contain the input image with the added mask of the detected blobs. Rectangles indicating the detected blobs were introduced during simulation for improved visibility.

objects to be detected; in our case, a square of size 4×4 pixels was appropriate. For every bright pixel, the structuring element is positioned in all the positions overlapping with the pixel and the element of the voting matrix is calculated. The voting matrix is finally compared with the constant threshold, as shown in (21):

$$m_V(x,y) = \begin{cases} 1 & \text{if } Hough_{4\times4}[\mathbf{m}_{BEHSX}]_{(x,y)} > \Theta_H \\ 0 & \text{otherwise} \end{cases} \quad (21)$$

It must be noted that the Hough transform has a tendency to connect the blobs which are very close together. However, this transformation significantly improves many others aspects of the final mask; the transformed blobs usually have a more convenient shape for labeling and speed estimation described in the next section, so the use of this transformation is very important to the overall efficiency of the algorithm.

4 Blob Analysis and Speed Estimation

The blobs obtained from the previously described blocks have to be analyzed to detect and to measure the speed of the moving vehicles. Since the camera usually observes the scene at some angle, additional transformations of the image are needed.

For speed estimation, knowledge regarding the relationship between the blobs' dimensions found on the image and the real world coordinates is necessary. Here, the authors assume a model like in [55], where the camera is located above the ground and is pointed towards the road. The ground level is assumed to be planar. Such a model is presented in Fig. 11. If we additionally assume that the observed objects are also planar, so their heights are $Z = 0$, then the X, Y coordinates on the road can be transformed into the x_{CAM}, y_{CAM} coordinates of the image on the camera sensor as [55]:

$$x_{CAM} = f\frac{X}{Y\cos(\varphi) + h/\sin(\varphi)} \quad (22)$$

$$y_{CAM} = f\frac{Y\sin(\varphi)}{Y\cos(\varphi) + h/\sin(\varphi)} \quad (23)$$

where:

- φ tilt angle [rad]
- h height of the camera above the road [m] and
- f focal length of the camera [m].

Thus, knowing the parameters f, h and φ, one can calculate the real world coordinates X, Y from the x_{CAM}, y_{CAM} coordinates. The input image is transformed into the image

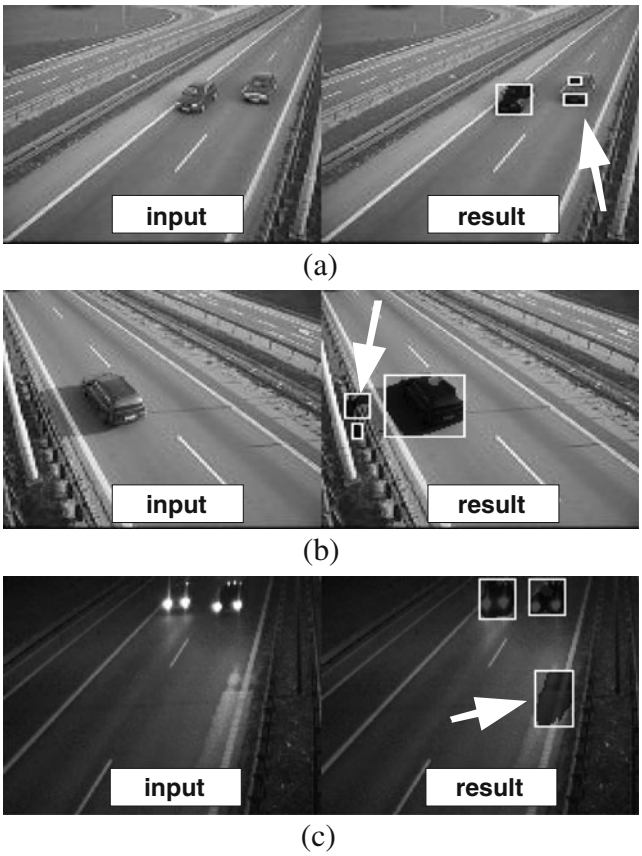

Figure 14 Simulation results of the algorithm, white arrows show the problematic situations for the detection algorithm (**a**)—a gray vehicle with minority of pixels detected, frame #1778 from "*obwodnica_4*" movie, (**b**)—the shadow detected as moving object in a sunny day, frame #42 from "*obwodnica_6*" movie, (**c**)—the highlight detected as moving object at night, frame #309 from "*obwodnica_noc*" movie. Frames "result" on the right contain the input image with the added mask of the detected blobs. Rectangles indicating the detected blobs were introduced during simulation for improved visibility.

which provides the linear correspondence between pixels x_{lin}, y_{lin} on the transformed image and real-world coordinates X, Y, where x_{map}, y_{map} are the indexes of pixels on the camera converter. An example of transformation is presented in Fig. 12. For a better view, the original image is presented instead of blobs.

The detected blobs are labeled and the following parameters are estimated: object's boundaries, centre of the object, area in pixels, fill factor (as the percentage of pixels with respect to the bounding rectangle area). The parameters are calculated using simple operations during pixel by pixel revision of the image. After this stage, a table with the column number equal to the number of indexed objects is created. Rows describe found parameters.

The objects which are small and have a small fill factor are discarded. The blobs which are overlapping on two subsequent frames are detected and marked. Such blobs are treated as the same object in movement. Estimation of the speed and direction of the objects is calculated by finding the distance between the centers of the objects marked in the previous stage.

5 Implementation Results

The algorithm described in the Section 3 has been tested with various video streams. The ground truth reference has been prepared for a set of video streams by manually extracting frame-by-frame all the pixels of each moving vehicle. The simulation results show that 57–94% of the pixels (depending on the stream) belonging to the moving vehicles in ground truth image are correctly identified by the algorithm (fill ratio *FIL* as defined in (6)). Moreover, the precision ratio *PR* defined in (7), indicating how many among the detected pixels belong to the moving objects, is about 56–88%. Simulation results for several frames of the selected video streams are shown in Fig. 13. As can be seen from Fig. 13, the algorithm is able to properly detect the moving vehicles at various scene conditions. A worse detection usually occurs in: dark scenes, for gray colored vehicles or in strong sun light causing intensive shadows. Such problematic situations with detection errors are collected in Fig. 14, the arrows indicate the erroneous detection results. The shapes of the resulting blobs in many situations are different from the shapes of the real moving objects, but for the purpose of simplified tracking and

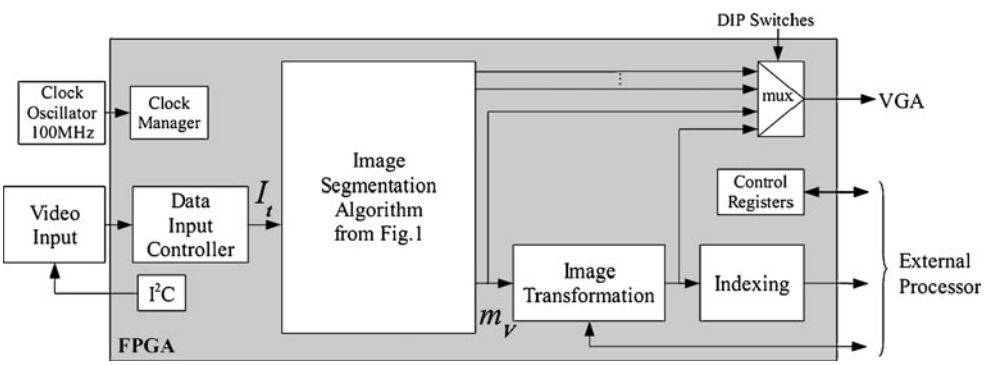

Figure 15 A simplified structure of the realized algorithm in FPGA.

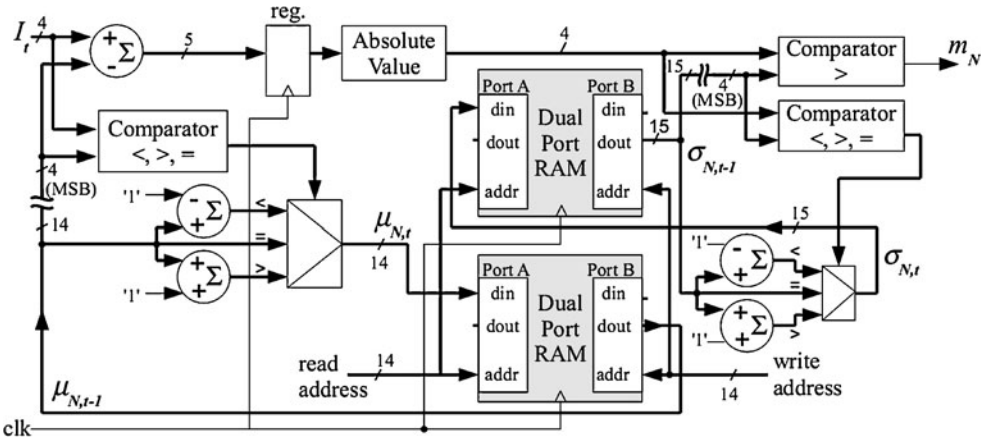

Figure 16 Simplified schematic diagram of non-selective background block implementation.

traffic measurement, the detailed shape is not very important.

The algorithm has been implemented in real hardware using Xilinx Virtex-4 SX FPGA on prototype board Virtex-4 Evaluation Kit from Avnet, utilizing approx. 1700 LUTs, 1200 flip-flops and 2.3 Mbits of the built-in RAM. The design has been written in VHDL and it has been synthesized and implemented using Xilinx's ISE 9.1.03i. The analog signal from the camera was being captured by Philips's SAA7113H video input processor. On-chip implementation included: selective background, non-selective background, background masks combination, temporal and spatial edge detection, highlight, shadow and extra pixel detection, Hough transform, geometrical transformation with indexing (i. e. blob labeling and blob parameter evaluation), as shown in Fig. 15. The interface to the external processor was used to collect the table with the detected objects' parameters.

The details of the implementation of the non-selective background update and subtraction is shown in Fig. 16. The values of μ_N and σ_N are stored in the dual port RAMs and are updated with every new pixel data. The selective background block is realized in a similar way, with the selectivity information added.

The implementation of the binary mask combination (Fig. 17) contains a shift register of a length $w + 1$, where w is the length of a single video line. The previously analyzed pixels, stored in the shift register, are used to calculate the mask $\mathbf{m}_B$. The similar shift registers have also been used for calculating the erosion and the dilation in masks $\mathbf{m}_{HS}$ and $\mathbf{m}_{BEHSX}$, the edges in the edge detection block and the indexes in the blob indexing block.

Due to the properties of the algorithm, all the remaining blocks from Fig. 1, except for the Hough block, are implemented in a similar way as the blocks shown in Figs. 16 and 17, requiring only a few cycles of the main 1.79 MHz clock to calculate the result and providing the possibility to obtain the pipelined implementation.

The Hough block requires, that for each pixel, a rectangular structuring element of a size of 4×4 is moved around the pixel and added to the voting matrix. Instead of

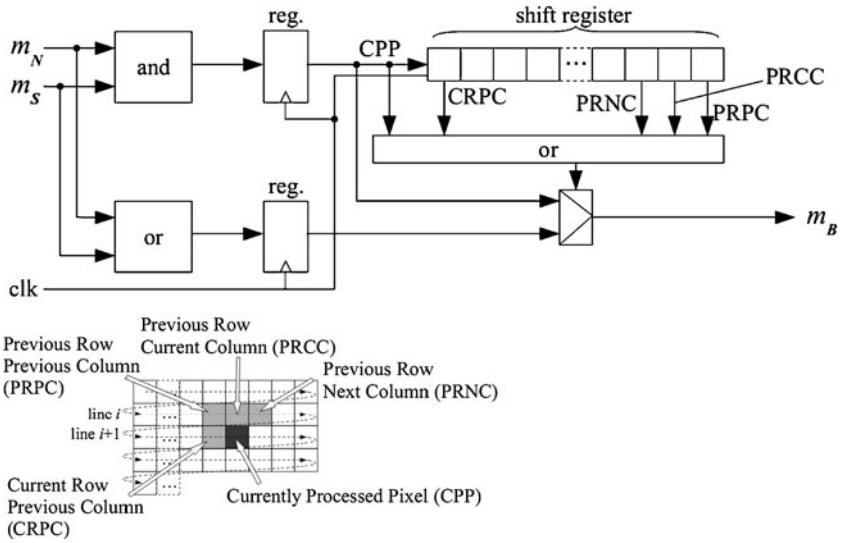

Figure 17 Simplified schematic diagram of the implementation of the binary mask combination.

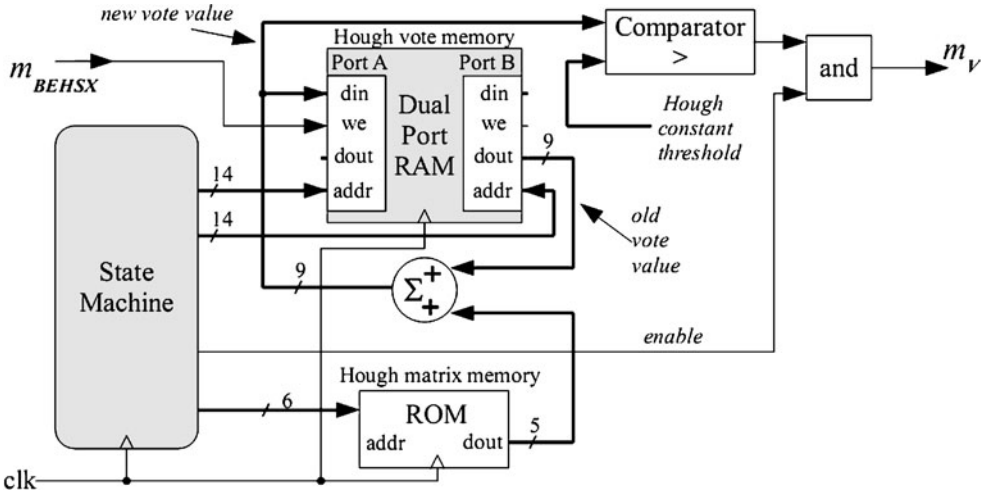

Figure 18 Simplified schematic diagram of the implemented Hough block.

moving the 4×4 structuring element, a 7×7 rectangular matrix of 5-bit values has been used. In this way, the pixel-centered matrix is stationary for every pixel and it is stored in the Hough matrix memory, as shown in Fig. 18.

The Hough block requires 49 clock cycles to calculate all the elements of the matrix, so the clock of the frequency of 28.5 MHz has been used for this block to provide coherent operation with the other blocks. The indexing block also works with the faster clock, as it also requires many internal iterations.

The image transformation block uses two memories for transforming the coordinates of each pixel of the input image. At the start of the system, the mapping memories should be programmed with the values calculated by the external processor. At normal operation, the transformation of single pixel takes only 3 cycles of the main clock for that block.

The system has been tested without image stabilization block, which is only needed in extreme situations, i.e. at a strong wind. The relative distribution of hardware resources among the blocks is shown in Table 2. The algorithm makes possible to use only integer values in all the calculations. All the constants required by the algorithm are read from the on-chip memory and are stored in the control registers. Table 2 also contains a typical number of clock cycles required to process a single pixel.

Table 2 Relative usage of hardware resources and relative power consumption.

Block	Look-Up Tables (LUT) [%]	Flip-Flops (FFs) [%]	Memory bits [%]	Clock cycles per pixel[a] (typ.)	Main clock frequency [MHz]	Relative Power [%]
Data Input Controller	11,2	7,5	6,3	10	25	5
Non-selective Background Update and Background Subtraction	6,9	7,1	23	2	1.79	16
Selective Background Update and Background Subtraction	7,6	6,4	18,3	2	1.79	12,8
Edge Detection (Temporal and Spatial)	11,1	5,5	4,1	2	1.79	4,3
Shadow Detection	3	1,1	0	2	1.79	0,3
Highlight and Extra-pixel Detection and Non-linear Brightness Transformation	2	1,2	0	2	1.79	0.3
Binary Mask Combination	0,2	0,8	0	2	1.79	0,1
Shadow and Highlight Processing	0,4	0,9	0	2	1.79	0,2
Final Processing	0,5	1,6	0	2	1.79	0,2
Hough Transform	5,3	9,4	7,9	49	28.5	8,7
Image Transformation	2,6	2,2	12,7	3	1.79	17,3
Indexing	23,7	18	15,1	12	28.5	18
Splice Table Generation	11,9	9,6	12,6	3	1.79	15,7
Control registers and glue logic	13,6	28,7	0	1	1.79	1,1

[a] Typical number of clock cycles per pixel is given, but some additional clock cycles at the moments between the picture frames can be used to finish the pending operations for some blocks. Data Input Controller works with the image of bigger resolution from the video source, so it has a larger number of clock cycles per pixel

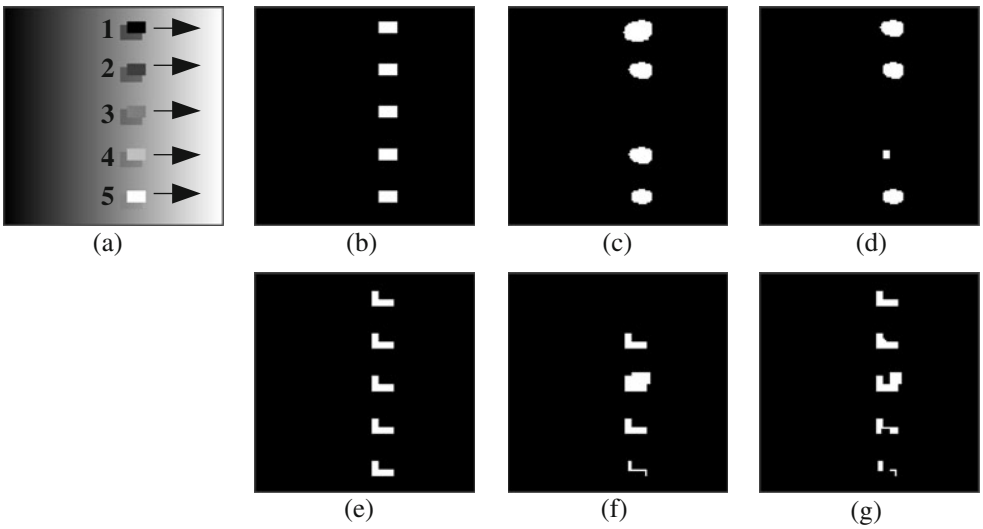

Figure 19 Simulation results of algorithm operation for artificial scene for 8- and 4-bit versions; (**a**)—input image with added objects' indices and arrows indicating direction of movement, (**b**)—ground truth for objects, (**c**)—object detection results (mask $\mathbf{m_V}$) for 8-bit version of the algorithm, (**d**)—object detection results (mask $\mathbf{m_V}$) for 4-bit version of the algorithm, (**e**)—ground truth for shadows, (**f**)—shadow detection results (mask $\mathbf{m_{SH}}$) for 8-bit version of the algorithm, (**g**)—shadow detection results (mask $\mathbf{m_{SH}}$) for 4-bit version of the algorithm.

All the simulation results presented in this paper have been done using 8-bit image representation. As already shown in Fig. 16, in the implemented system only the 4 most significant bits have been used, which was forced by the limited resources of the FPGA. To show the influence of this reduction, the simulation has been made using artificial test scene of linearly changing background of μ_A= 0…255, with rectangular objects casting simulated shadows of intensity $\gamma_k\mu_A$, moving from left to right, as shown in Fig. 19. The parameters of the artificial moving objects are shown in Table 3.

To better illustrate the influence of data reduction, the FIL and PR ratios have been calculated and presented in Fig. 20. As can be seen from the simulations, data reduction resulted in a lower sensitivity of the algorithm in detecting the objects and shadows. However, the relative precision of the 4-bit version of the algorithm (PR ratios) slightly has increased. The reduction of data width saved a lot of FPGA resources and power, at the expense of the decreased sensitivity. Nevertheless, the algorithm results still seem to be sufficient for this application.

The photographs showing the results of the algorithm are presented in Fig. 21. More results are available on-line at http://www.ue.eti.pg.gda.pl/sn. The hardware was designed to work with the main 1.79 MHz clock and the additional 28.5 MHz clock for the Hough and indexing blocks to process 25 frames per second of a low resolution 128×128 pixels monochrome image. The main clock frequency has been set as low as possible to enable the processing of pixel data by each block. The estimated dynamic power consumption was about 600 mW with 600 mW of quiescent power. The core elements realizing the algorithm were estimated to consume 400 mW, this power is distributed among the blocks as shown in the last column of Table 2. Since the FPGAs are known to have a large power demand, implementing the algorithm in ASIC would further decrease the power consumption. The obtained maximum clock frequency was about 135 MHz, which would permit the processing of up to 117 fps—this indicates the great potential of increasing the processing speed of the algorithm, for example higher resolutions of input video stream can easily be used. As can be seen from Table 2, the limiting stages for the algorithm are the Hough block and the indexing block. The authors decided to use the generalized Hough transform due to its property to detect the rectangular objects, but for simpler implementations, the morphological operations could also be used. The state machine in the indexing block requires many clock cycles per pixel to communicate with its memories and to index the blobs.

To demonstrate the efficiency of the FPGA realization, the software implementation in C of the same algorithm has been developed. The software version permitted the processing of about 160 fps using Intel's dual core processor with 2.13 GHz clock and maximum power dissipation of 65 W with Linux operating system—the speed is similar, but the FPGA implementation uses much less power.

The comparison of some parameters of the proposed implementation with the solution presented in [39] is shown in Table 4. The implementation described in [39] uses monochrome images of VGA resolution and the segmentation algorithm is based on MOG method, which should work better for non-stationary backgrounds. The implementation presented in this paper works with lower resolution images, but additionally contains the geometrical image transformation block, moreover, the highlight and shadow detection

Table 3 Parameters of the objects in the artificial test scene.

object's id k=	1	2	3	4	5
object's intensity	0	64	128	192	255
shadow coefficient γ_k	0.55	0.65	0.75	0.85	0.95

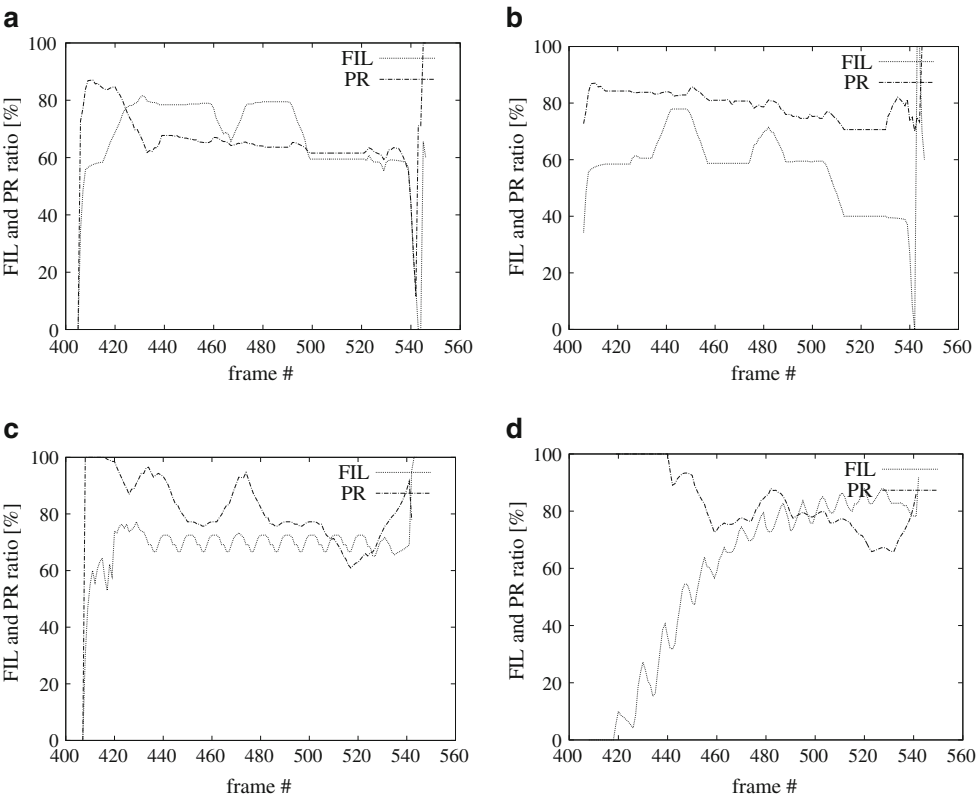

Figure 20 Calculated values of FIL and PR ratios for object and shadow detection in the artificial scene from Fig. 19; (**a**)—FIL and PR rations for objects using 8-bit data representation, (**b**)—FIL and PR rations for shadows using 8-bit data representation, (**c**)—FIL and PR rations for objects using 4-bit data representation, (**d**)—FIL and PR rations for shadows using 4-bit data representation.

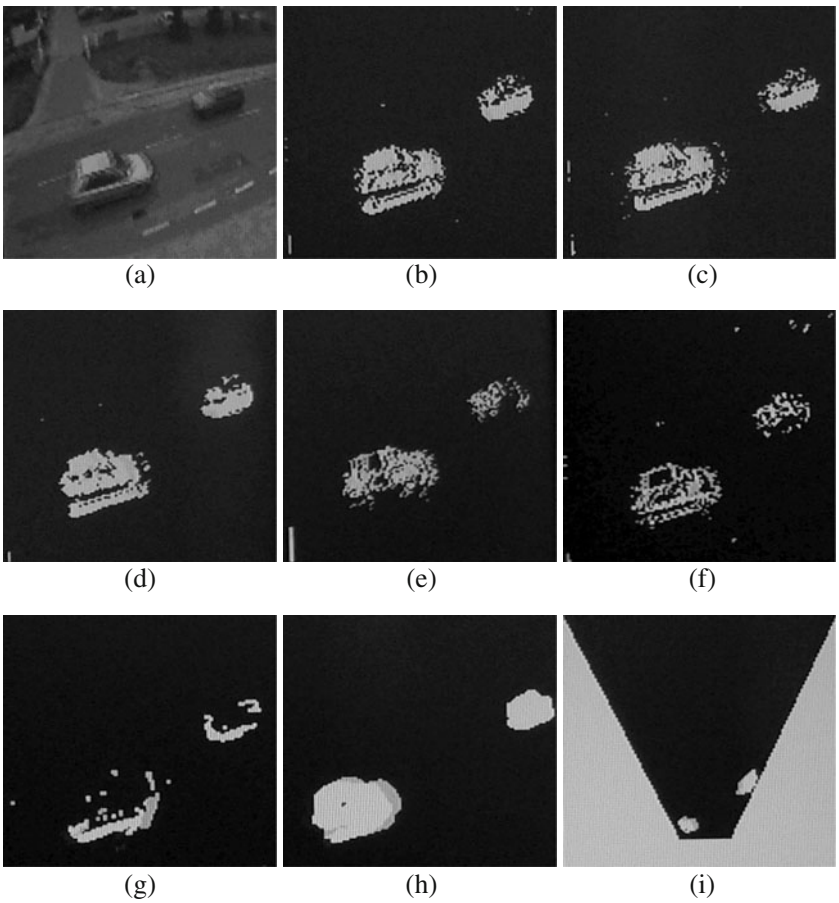

Figure 21 Photo of implementation results of the algorithm (**a**)—input image I_t, (**b**)—non-selective background mask m_N, (**c**)—selective background mask m_S, (**d**)—combined background mask m_B, (**e**)—temporal edge mask m_{ET}, (**f**)—spatial edge mask m_{ES}, (**g**)—shadow and highlight mask m_{HS}, (**h**)—mask m_V after final processing, (**i**)—mask after geometrical transformation for ϕ=29.8º, h=7 m, f=2.8 mm.

Table 4 The comparison of the selected design parameters of the algorithm with the implementation presented in [39].

Parameter	The implementation presented in [39]	The implementation presented in this paper	
		Full version	Speed optimized version
Image resolution [pixels]	640×480	128×128	128×128
Clock frequency [MHz]	65	Max. 135	Max. 150
Frame processing speed [fps]	35	Max. 117	Max. 2528
processing speed [Mpixels/s]	10.7	1.9	41.4
FFs used	1,316 (4.3%[a])	1,200 (3.9%[a])	891 (2.9%[a])
LUTs used	1,232 (4.0%[a])	1,700 (5.5%[a])	1,683 (5.4%[a])
Block RAMs [Mbits]	0.34 (9.9%[a])	2,3 (66.7%[a])	1.6 (46.4%[a])
External RAM [Mbits]	20	–	–
Implemented features	background subtraction, MOG, connected component labeling	selective background subtraction, non selective background subtraction, spatial edge detection, temporal edge detection, shadow detection, highlight detection, Hough transform, geometrical image transform, connected component labeling	selective background subtraction, non selective background subtraction, spatial edge detection, temporal edge detection, shadow detection, highlight detection, geometrical image transform, connected component labeling (1st phase only with label equivalence table generation)

[a] Relative to XilinxVirtex-4 XC4VSX35 FPGA

blocks together with the edge detection blocks should provide a better detection sensitivity. To show the potential speed of the presented algorithm, the speed-optimized version of reduced functionality has also been included in the comparison in Table 4. In the speed optimized version, the Hough block is removed and the indexing block is reduced to the 1st phase of the connected component algorithm with the label equivalence table generation. No power information is given in [39], so it has not been compared.

6 Conclusions

In this paper, the combined algorithm for extracting moving objects from a real-time video stream is proposed. The processing steps were carefully selected and adopted to provide simple and straightforward realization in specialized hardware, such as FPGA or ASIC. A few novel ideas to enhance the algorithm are also developed, increasing the robustness and maintaining its simplicity for hardware implementation. The proposed method of background calculation, using running mode is very fast and requires only basic operations. The novel combination of masks from selective and non-selective backgrounds improves the detection quality. The non-linear brightness transformations enable correct detection of shadows and highlights in various light conditions. The further improvement could include the automatic recognition of day and night with switching between shadow and highlight detection. The application of generalized Hough transformation significantly improves the final blob mask. However, to simplify the hardware, the Hough block could be replaced with a set of morphological operations. The proposed algorithm has been implemented in FPGA and tested in the real environment. The test results proved the usability of the presented idea for recognizing the moving vehicles at low power consumption—the system properly found almost all of the moving vehicles during the day and night.

Acknowledgments This work was supported in part by the Polish Ministry of Science and Higher Education under R&D grant no. R02 014 01.

References

1. Tan, T. N., & Baker, K. D. (2000). Efficient image gradient based vehicle localization. *IEEE Transactions on Image Processing, 9* (8), 1343–1356.
2. Rajagopalan, A. N., & Chellappa, R. (2000). Vehicle detection and tracking in video. In *Proc. Int. Conf. Image Process,* Vancouver, BC, Canada, vol. 1, pp. 351–354.
3. Koller, D., Weber, J., & Malik, J. (1994). Robust multiple car tracking with occlusion reasoning. In *Proc. European Conf. On Computer Vision*, Stockholm, Sweden, pp. 189–196.
4. Cheung, S. C., & Kamath, C. (2004). Robust techniques for background subtraction in urban traffic video. In *Proc. SPIE Video Communications and Image Processing*, pp. 881–892.
5. KaewTraKulPong, P., & Bowden, R. (2003). A real time adaptive visual surveillance system for tracking low-resolution colour targets in dynamically changing scenes. *Image and Vision Computing, 21*(10), 913–929.
6. Wojcikowski, M., Zaglewski, R., & Pankiewicz, B. (2008). An intelligent image processing sensor—the algorithm and the hardware

implementation. In *Proc. Int. Conf. Information Technology*, Gdansk, Poland.
7. Cucchiara, R., Grana, C., Piccardi, M., & Prati, A. (2000). Statistic and knowledge-based moving object detection in traffic scenes. In *IEEE Proc. Intelligent Transportation Systems*, Dearborn, MI, pp. 27–32.
8. Cucchiara, R., Granna, C., Piccardi, M., & Prati, A. (2003). Detecting moving objects, ghosts and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25*(10), 1337–1342.
9. Gorgon, M., Pawlik, P., Jablonski, M., & Przybylo, J. (2007). FPGA-based road traffic videodetector. In *Proc. 12th Euromicro Conf. on Digital System Design, Architectures, Methods and Tools (DSD '09)*, Lubeck, Gernamy.
10. Pamula, W. (2009). Vehicle detection algorithm for FPGA based implementation. *Computer Recognition Systems 3*, Springer Berlin/Heidelberg, vol. 57/2009, pp. 585–592.
11. Silberberg, T. M., Harwood, D. A., & Davis, L. S. (1986). Object recognition using oriented model points. *Computer Vision, Graphics, and Image Processing, 35*(1), 47–71.
12. Horaud, R. (1987). New methods for matching 3D objects with single perspective view. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 9*(3), 401–412.
13. Lowe, D. G. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence, 31*(3), 355–395.
14. Koller, D., Daniilidis, K., & Nagel, H.-H. (1993). Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision, 10*, 257–281.
15. Grimson, W. E. L. (1990). The combinatorics of object recognition in cluttered environments using constrained search. *Artificial Intelligence, 44*(1–2), 121–165.
16. Horn, B. K. P., & Schunk, B. G. (1981). Determining optical flow. *Artificial Intelligence, 17*, 185–203.
17. Barron, J. L., Fleet, D. J., & Beauchemin, S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision, 12*(1), 43–77.
18. Bainbridge-Smith, A., & Lane, R. G. (1997). Determining optical flow using a differential method. *Image and Vision Computing, 15*(1), 11–22.
19. McIvor, A. M. (2000). Background subtraction techniques. In *Proc. of Image and Vision Computing*, Auckland, New Zealand.
20. Piccardi, M. (2004). Background subtraction techniques: A review. In *Proc. IEEE Conf. Syst., Man Cybern.*, Hague, The Netherlands, pp. 3099–3104.
21. Bayona, A., SanMiguel, J. C., & Martínez, J. M. (2009). Comparative evaluation of stationary foreground object detection algorithms based on background subtraction techniques. *Advanced Video and Signal Based Surveillanc*e, 2009, AVSS '09, Sixth IEEE International Conference on, pp. 25–30, doi:10.1109/AVSS.2009.35.
22. Benezeth, Y., Jodoin, P. M., Emile, B., Laurent, H., & Rosenberger, C. (2008). Review and evaluation of commonly-implemented background subtraction algorithms. In *Proc. 19th Int. Conf. Pattern Recognition, ICPR*, pp. 1–4.
23. Grove, T. D., Baker, K. D., & Tan, T. N. (1998). Colour based object tracking. In *Proc. 14th Int. Conf. Pattern Recognition*, Brisbane, Qld., Australia, pp. 1442–1444.
24. Rittscher, J., Kato, J., Joga, S., & Blake, A. (2000). A probabilistic background model for tracking. In *Proc. European Conf. Comput. Vision*, pp. 336–350.
25. Yang, Y. H., & Levine, M. D. (1992). The background primal sketch: an approach for tracking moving objects. *Machine Vision and Applications, 5*(1), 17–34.
26. Cucchiara, R., Granna, C., Piccardi, M., Prati, A., & Sirotti, S. (2001). Improving shadow suppression in moving object detection with HSV color information. In *Proc. IEEE Intell. Transp. Syst. Conf.*, Oakland, CA, pp. 334–339.
27. McFarlane, N., & Schofield, C. (1995). Segmentation and tracking of piglets in images. *Machine Vision and Applications, 8*(3), 187–193.
28. Wren, C., Azarbayejani, A., Darrell, T., & Pentland, A. (1997). Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*(7), 780–785.
29. Franois, A., & Medioni, G. (1999). Adaptive color background modeling for real-time segmentation of video streams. In *Proc. of Int. Conf. Imaging Science, Systems and Technology*, Las Vegas, NV, pp. 227–232.
30. McKenna, S. J., Jabri, S., Duric, Z., & Rosenfeld, A. (2000). Tracking groups of people. *Computer Vision and Image Understanding, 80*(1), 42–56.
31. Haritaoglu, I., Harwood, D., & Davis, L. S. (2000). W4: real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 809–830.
32. Zhang, R., Zhang, S., & Yu, S. (2007). Moving objects detection method based on brightness distortion and chromaticity distortion. *IEEE Transactions on Consumer Electronics, 53*(3), 1177–1185.
33. Grimson, W. E. L., Stauffer, C., Romano, R., & Lee, L. (1998). Using adaptive tracking to classify and monitor activities in a site. In *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, Santa Barbara, CA, pp. 22–29.
34. Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *Conf. Computer Vision and Pattern Recognition CVPR'99*, vol. II, pp. 246–252.
35. Stauffer, C., & Grimson, W. E. L. (2000). Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 747–757.
36. KaewTraKulPong, P., & Bowden, R. (2001). An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proc. 2nd European Workshop on Advanced Video-based Surveillance Systems,* Kingston upon Thames.
37. Ren, Y., Chua, C., & Ho, Y. (2003). Motion detection with non-stationary background. *Machine Vision and Applications, 13*(5–6), 332–343.
38. Javed, O., Shafique, K., & Shah, M. (2002). A hierarchical approach to robust background subtraction using color and gradient information. In *IEEE Workshop Motion Video Computing., IEEE Comput. Soc.*, Orlando, FL, pp. 22–27.
39. Appiah, K., Hunter, A., Dickinson, P., & Meng, H. (2010). Accelerated hardware video object segmentation: From foreground detection to connected components labeling. *Computer Vision and Image Understanding*. doi:10.1016/j.cviu.2010.03.021.
40. Elgammal, A., Harwood, D., & Davis, L. S. (2000). Non-parametric model for background subtraction. In *European Conf. Computer Vision*, Dublin, Ireland, vol. II, pp. 751–767.
41. Toyama, K., Krumm, J., Brumitt, B., & Meyers, B. (1999). Wallflower: Principles and practice of background maintenance. In *Proc. ICCV99*, Corfu, Greece, pp. 255–261.
42. Comaniciu, D., & Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(5), 603–619.
43. Duque, D., Santos, H., & Cortez, P. (2005). Moving object detection unaffected by cast shadows. Highlights and ghosts. In *Proc. IEEE Int. Conf. Image Processing*, pp. 413–416.
44. Porikli, F., Ivanov, Y., & Haga, T. (2008). Robust abandoned object detection using dual foregrounds. *EURASIP Journal on Advances in Signal Processing*. doi:10.1155/2008/197875.
45. PETS'2001, Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, December 9, 2001.
46. Karaman, M., Goldmann, L., Yu, D., & Sikora, T. (2005). Comparison of static background segmentation methods. In *Proc. of SPIE*, vol. 5960, pp 2140–2151.

47. Cai, J., Shehata, M., & Badawy, W. (2009). A robust video-based algorithm for detecting snow movement in traffic scenes. *Journal of Signal Processing Systems, 56*(2–3), 307–326. New York: Springer.
48. Scanlan, J. M., Chabries, D. M., & Christiansen, R. W. (1990). A shadow detection and removal algorithm for 2-D images. In *Proc. Int. Conf. Acoustic Speech Signal Process.*, IEEE, Albuquerque, NM, pp. 2057–2060.
49. Mikic, I., Cosman, P. C., Kogut, G. T., & Trivedi, M. M. (2000). Moving shadow and object detection in traffic scenes. In *Proc. 15th Int. Conf. Pattern Recog.*, Barcelona, Spain, vol. 1, pp. 321–324.
50. Stauder, J., Mech, R., & Ostermann, J. (1999). Detection of moving cast shadows for object segmentation. *IEEE Transactions on Multimedia, 1*(1), 65–76.
51. Johnston, C. T., Gribbon, K. T., & Bailey, D. G. (2004). Implementing image processing algorithms on FPGAs. In *Proceedings of the Eleventh Electronics New Zealand Conference (ENZCON'04)*, Palmerston North.
52. Levine, B., Colonna, B., Oblak, T., Hughes, E., Hoffelder, M., & Schmit, H. (2003). Implementation of a target recognition application using pipelined reconfigurable hardware. In *International Conference on Military and Aerospace Applications of Programmable Devices and Technologies*.
53. Horprasert, T., Harwood, D., & Davis, L. (1999). A statistical approach for real-time robust background subtraction and shadow detection. In *IEEE Frame-Rate Applicat. Workshop*, Kerkyra, Greece.
54. Prati, A., Mikić, I., Trivedi, M., & Cucchiara, R. (2003). Detecting moving shadows: algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25*(7), 918–923.
55. Schoepflin, T. N., & Dailey, D. J. (2003). Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems, 4*(2), 90–98.

Marek Wójcikowski received his M. Sc. degree in Electrical Engineering in 1993 from the Technical University of Gdansk, Poland. In 1994 he was on leave at the University of Karlsruhe, Germany. In 2002 he received his PhD degree in Electrical Engineering from the Technical University of Gdansk, Poland. Since 1994 he has been with the Department of Microelectronic Systems, Technical University of Gdańsk, Poland. His research interests lie in design of microelectronic systems and sensor networks.

Robert Żaglewski received his M. Sc. degree in Electrical Engineering in 2007 from the Technical University of Gdansk, Poland. In years 2007-2008 he was with the Department of Microelectronic Systems, Technical University of Gdańsk, Poland. He is now Intel Shannon Ltd, Shannon, Ireland. His research interests lie in FPGA and ASIC design.

Bogdan Pankiewicz received his M. Sc. degree in Electrical Engineering in 1993 from the Technical University of Gdansk, Poland. In 2002 he received his PhD degree in Electrical Engineering from the Technical University of Gdansk, Poland. Since 1994 he has been with the Department of Microelectronic Systems, Technical University of Gdańsk, Poland. His research interests lie in design of analog and digital integrated circuits.