# Hardware-software implementation of vehicle detection and counting using virtual detection lines

Tomasz Kryjak, Mateusz Komorkiewicz, Marek Gorgon *Senior Member IEEE*
AGH University of Science and Technology,
Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering,
Department of Automatics and Biomedical Engineering,
al. Mickiewicza 30, 30-059 Krakow, Poland

***Abstract*—In this paper a hardware-software system for vehicle detection and counting at intersections is presented. It has been implemented and evaluated on the heterogeneous Zynq platform. The vehicle detection is based on the concept of virtual detection lines (VDL). Differences in colour, horizontal edges and Census transform results between areas around the VDL for two consecutive frames are used to disclosure particular vehicles. This part of the system is designed in hardware description language and implemented in the reconfigurable resources of the Zynq platform. Information about the vehicle presence, along with the time-spatial image obtained at the VDL are transmitted and processed on the ARM core integrated in the Zynq device. This allows to perform further analysis and eliminate false or multiple detections. The solution has been evaluated on several video sequences recorded in various conditions: sunny day (the occurrence of deep shadows), cloudy day, rain and night-time. The system could be an important element of an intelligent transportation system (ITS) i.e. applied in a smart camera.**

## I. Introduction

The use of vision systems in traffic analysis and control (so-called Intelligent Transportation Systems – ITS) is becoming more widespread. This is evidenced by the increasing number of cameras installed near the roads. They are used to analyse and control the traffic at the intersections, detect traffic congestions, detect unusual events (e.g. collisions or accidents), estimate travel time between two cities, measure average speed on a given road section and as a part of toll collection systems (car parks, motorways).

The first of the mentioned applications seems to be especially important. A vision system mounted at the intersection can provide a range of relevant information. Firstly, it can be used as a virtual induction loop, which allows to detect the presence of vehicles. On this basis it is also possible to determine the vehicles queue length (equivalent of several loops at different distances from the traffic lights), the average speed of vehicles and traffic density. Alternative solutions: induction loops, passive magnetic sensors or pneumatic tubes have a common drawback. Their installation and maintenance in case of a failure requires interference with the road surface, which can be quite costly. In addition, the obtained information is only binary (vehicle / no vehicle). On the other hand, these solutions are not sensitive to external conditions (time of the day, weather conditions, etc.).

A vision based system also allows for: rough classification of vehicles (e.g. bikes / motorbikes, cars, minibuses, buses, trucks) [1], tracking vehicles between intersections (using the plate number or other features) and detection of abnormal situations (accidents, breakdowns, etc.). In certain conditions exact classification of vehicles (make and model) is also possible [2]. Furthermore, the human operator of the ITS system is able to download images from the camera and accurately assess the current situation. It is also noteworthy to point out certain disadvantages of a vision based system. First, this solutions are often affected by lighting and meteorological conditions. Good examples are shadows, heavy rain, snowfall or fog. Great challenges are also: the huge variety of vehicles (different sizes and shapes) and the limited computational power (efficiency of vision systems).

In the case of vision systems, the required calculations can be implemented in two variants. In the first, the video stream from cameras mounted at intersections is transmitted to the surveillance centre, where it is subjected to automatic or sometimes manual analysis. The main disadvantage is the need for very high bandwidth of communication infrastructure. In the second, the smart camera [3] concept is used. In this case, image processing and analysis are carried out immediately after image acquisition and there is no need to transmit every frame to other components of the system. Usually, the output contains only simple data like the number of detected vehicles (meta-data stream). Of course, the smart camera should also allow to access the raw video stream as this can be useful in analysis of unusual situations or system debug.

When designing a smart camera, a very important issue is the choice of the hardware platform. There are solutions based on general-purpose processors (CPU), digital signal processors (DSP) and reconfigurable devices (FPGA). To perform real-time image processing and analysis the third platform seems to be very attractive. In recent years it has been proved that reconfigurable systems can handle with many vision algorithms such as various filtration methods, complex background modelling and foreground object segmentation, optical flow computation, tracking and object classification systems (e.g. detection of human silhouettes) [4]. Many image processing systems implemented in FPGAs involve the pipeline data processing approach, where the pixel stream passes through different computing elements.

However, for some complex vision algorithms the pipeline implementation proved to be quite cumbersome or even impossible. An example is region growing segmentation, which requires an impossible to predict pixel access pattern. In such cases, the use of a general purpose processor system is a much more convenient solution. Modern FPGAs allow to use a so-called soft-processor (MicroBlaze from Xilinx, Nios from Altera), but these solutions have quite limited computing performance. In 2012 Xilinx introduced the Zynq heterogeneous platform, which is a combination of FPGA logic resources and the dual-core ARM processor (a similar platform is also available from Altera). The solution has gained some interest in the image processing community, as evidenced by a number of described applications: image filtering [5], ultrasound data processing [6], optical flow computation [7] or road sign recognition [8].

In this paper the concept of using Zynq device in a smart camera for intelligent transportation systems is considered. The implementation of a vehicle detection and counting algorithm is divided between the reconfigurable resources and ARM processor system with PetaLinux. To our best knowledge this is the first approach of using Zynq in a smart camera for ITS described in the literature. In Section II different approaches to vehicle detection based on visual information are described. The proposed algorithm and its evaluation are presented in Section III. Hardware-software implementation of the method is described in Section IV. Further research possibilities are discussed in Section V. The article ends with a summary.

## II. Video Based Systems for Vehicle Detection and Counting

Vehicle detection can be performed using background modelling and subtraction (i.e. foreground object detection) or optical flow based moving object segmentation. In the first case, objects are extracted using the differential image between the current frame and the so-called background model. This approach has been used, among others, in works [9] and [10]. Unfortunately, this solution has a number of drawbacks that hinder its practical application in traffic monitoring: low resistance to camera jitter (it is usually necessary to implement some kind of compensation algorithm), difficulties in proper initializing and reinitializing the background model (especially in the presence of heavy traffic), high sensitivity to shadows and sudden illumination changes (e.g. reflections of car lights on the road). In addition, the specific conditions present at an intersection cause that background elements (i.e. carriageway) are covered by cars waiting for the green light for extended periods of time. This significantly hinders the background modelling and can be the cause of many segmentation errors.

Moving object detection using optical flow or, in a simplified case, consecutive frame differencing, was applied to vehicle segmentation in works [11] and [12]. The solution helps to eliminate some of the disadvantages characteristic to background subtraction (e.g. sensitivity to camera movement, problems with maintaining the correct background model), but only allows to segment vehicles that are moving. This greatly complicates the analysis of the situation on an intersection. Furthermore, the obtained object masks often require complex post-processing, since the optical flow field for homogeneous areas is usually incorrect (e.g. division of a large object). In the literature, also more advanced solutions, such as using 3D deformable models [13] are described. However, the computational complexity limits their usage in embedded devices.

A very interesting approach, that is frequently used in recent research papers, are virtual detection lines (VDL) and time-spatial images (TSI). The idea is presented in Figure 1. The basis is a virtual detection line (VDL) located on a given part of the road (red line). In each frame, the pixels on the VDL are stored in a buffer and form the time-spatial image. The TSI image contains informations about vehicles width (x-axis) and vehicle size/speed (t-axis). This can be used to implement vehicle counting, speed estimation or classification [1], [14].

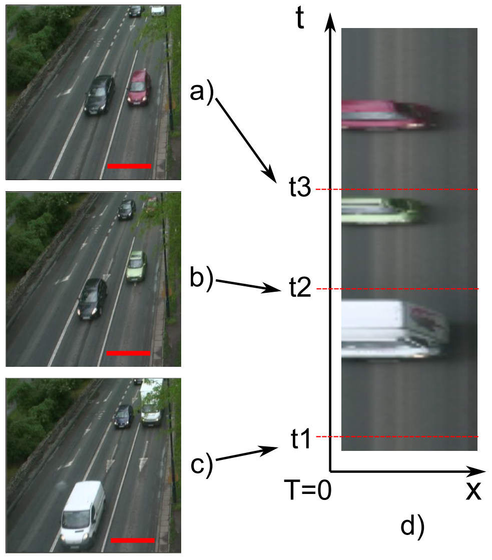

Fig. 1. The idea of using virtual detection lines (VDL) and time-spatial images (TSI). a) frame at time $t3$, b) frame at time $t2$, c) frame at time $t1$, d) vehicles on the TSI, x-axis – spatial, t-axis – time.

In the literature two approaches are presented. The TSI image is generated based on the object mask obtained using background subtraction [1] or directly using the raw image from the camera [14]. In the second case, edge detection (e.g. Canny algorithm) followed by some morphological post-processing (closing, filling holes) is often applied. This allows to obtain masks of individual objects (vehicles). It is also possible to integrate the results from several VDL/TSI, which can improve the reliability of the system [14]. An important feature of this approach is also the relatively low computational complexity due to processing only a rather small portion of the image. Additionally, a system with many VDLs can be easily realised in a parallel computing architecture, such as FPGA devices.

Hardware implementations of vehicle detection algorithms have been described in several papers. One of the first works [15] used a relatively simple background model and the SAD algorithm. To maintain a proper background model, the update was performed only when no vehicles were detected at the specific location. The system was implemented in Handel-C HLS language and PixelStream library. It was evaluated on RC300E platform with a Virtex II FPGA device. It allowed to process 25 frames with 786 $\times$ 576 pixels resolution per second.

In the article [16] a vehicle detection system that is based on the analysis of edges and 3D images was presented. The proposed algorithm requires, among others, histogram equalization, Canny edge detection, corner detection, vehicle corner region expansion and vehicle validation modules. The system was implemented on a platform equipped with a Virtex 5 FPGA and is able to process 60 frames with a resolution of $640 \times 480$ per second.

In the paper [17] the hardware implementation of a traffic analysis algorithm was presented. The segmentation is based on background modelling and subtraction (short and long-term background models), supplemented by shadows and light reflections detection. The application allows to measure the speed of vehicles. To increase reliability, a geometric transformation of the image is applied. The system has been evaluated on a Virtex 4 FPGA and allows to process 32 frames with a resolution of $128 \times 128$ per second. The correct detection rate of the system given by the authors was 90% at day and 56% at night (100 cars in each test).

In articles [18] and [19] extended versions of the above described system were presented. Edges were added to the background model, the generalized Hough transform was applied and object tracking based on a binary mask was implemented (executed in the CPU). A very valuable part of the work is its practical verification in urban conditions. A total number of 26 nodes was used (22 based on FPGA and 4 on ASIC). The authors report an accuracy of 93% at a sunny day, 83% at a cloudy day and 63% at night (100 cars in each test).

## III. The Proposed Algorithm

Several factors were taken into account while designing the algorithm: accuracy, computational complexity, the possibility to divide the computations between hardware and software, resistance to various lightning conditions (time of the day, shadows) and camera jitter, as well as the ability to work in conditions occurring at a crowded intersection. The last factor requires an extended comment. Some approaches described in the literature [20], [14], [16] were designed and tested on sequences recorded by cameras mounted over a road, where the vehicle movement is usually smooth (e.g. highway). In such cases methods based on background subtraction or optical flow can obtain quite good results.

In this work it was assumed that the system should be able to operate at a typical intersection, where many cars stop at red light. This condition, as well as the characteristics described above, led to the conclusion that background modelling or optical flow methods are not the adequate solution for the designed application. It was therefore decided to use an approach based on virtual detection lines (VDL) and time-spatial images (TSI).

To reduce the impact of external lighting conditions and camera jitter, all operations were performed on two consecutive frames. The concept is based on detecting the presence of vehicles by analysing only the local neighbourhood of a VDL – this issue is described in details is Subsection III-A. As a result small images (patches) containing vehicles are obtained. In the second stage, the patches are analysed to eliminate erroneous (caused by shadow or other disturbances) or multiple detections – a detailed description is provided in Subsection III-B. The idea is schematically shown in Figure 2. The evaluation of the proposed solution is presented in Subsection III-C. It should be noted that the method is a development of the approaches described in the literature, particular in the works [1] and [14].

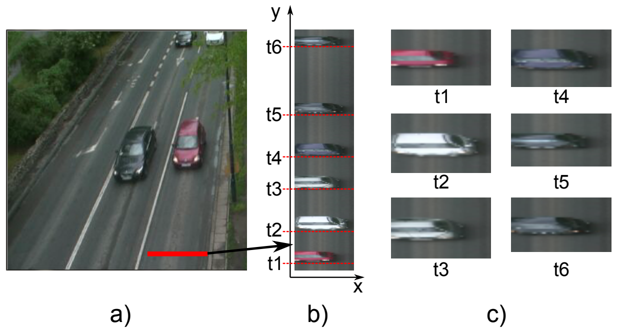

Fig. 2. Scheme of the proposed vehicle detection and counting system. a) VDL located on the road, b) TSI, c) obtained patches

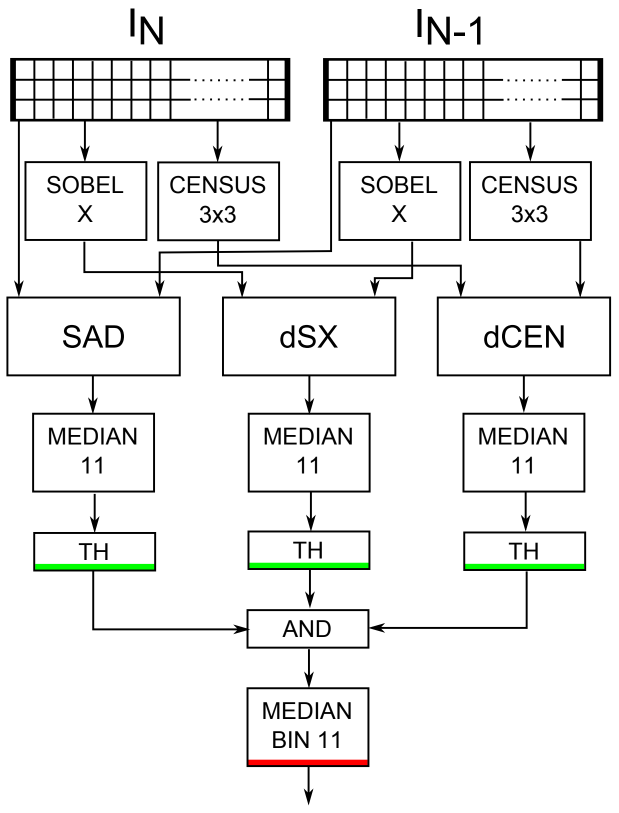

Fig. 3. Scheme of the proposed vehicle presence detection algorithm

### A. Vehicle Presence Detection

The vehicle presence detection is based on identifying similarities between two successive (in terms of time) VDL neighbourhoods of size 3 $\times$ VDL width. Scheme of the proposed solution is presented in Figure 3. $I_N$ and $I_{N-1}$ stand for the current and previous VDL context. In the first step horizontal Sobel gradient (*SOBEL X*) and $3 \times 3$ Census transform (*CENSUS 3x3*) are computed. The Census transform is defined by the following equation:

$$C(\omega) = \begin{cases} 1 & \text{if } IG(\omega) > IG(c) + \text{bias} \\ 0 & \text{else} \end{cases} \quad (1)$$

where: $\omega$ – $3 \times 3$ context without the central pixel, $IG(\omega)$ – pixel from $\omega$ context (image in greyscale), $c$ – centre of the

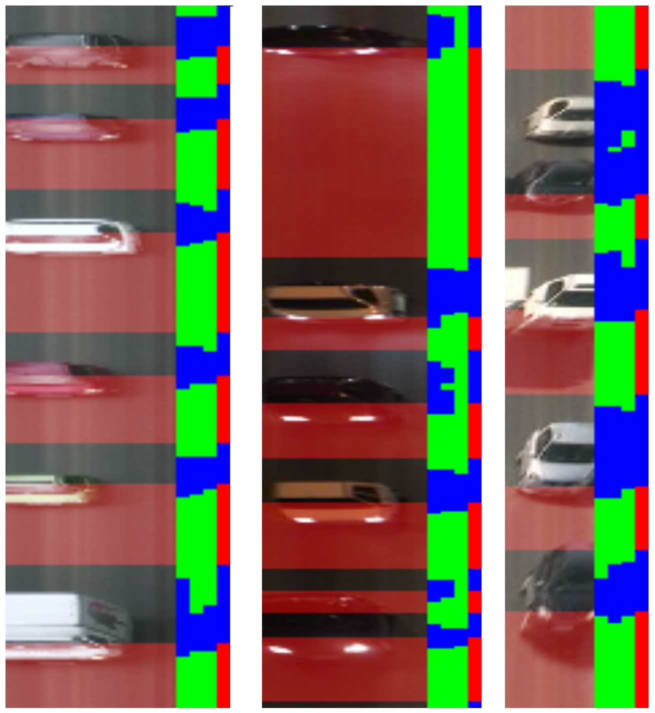

Fig. 4. Sample results of the proposed vehicle presence detection method

$\omega$ context, $IG(c)$ – central pixel, $bias$ – additional bias to the binarization threshold. The result of the operation described by Equation (1) forms an 8-bit vector $C$.

As similarity measures are used:

- SAD:
$$SAD = \sum_{i=0}^{3}\sum_{j=0}^{L}\sum_{k=0}^{3} |I_N^k(i,j) - I_{N-1}^k(i,j)| \quad (2)$$
where: $i$ – line index in the VDL neighbourhood (vertical), $L$ – width of the detection line, $I_N$ – $N$-th image frame in RGB colour space, $k$ – particular colour component $\{$R,G,B$\}$.
- absolute value of differences between edge images (computed using horizontal Sobel gradient):
$$dSX = \sum_{i=0}^{3}\sum_{j=0}^{L}\sum_{k=0}^{3} |SX_N^k(i,j) - SX_{N-1}^k(i,j)| \quad (3)$$
where: $SX_N$ – horizontal Sobel gradient for $N$-th video frame.
- Hamming distance computed for binary $3 \times 3$ Census transform result obtained for consecutive frames:
$$dCEN = \sum_{i=0}^{3}\sum_{j=0}^{L}\sum_{q=0}^{8} C_N(i,j,q) \;\; XOR \;\; C_{N-1}(i,j,q) \quad (4)$$
where: $C_N$ – Census transform result for $N$-th frame, $q$ – index in an 8-bit vector (result of the Census transform for a $3 \times 3$ window).

The resulting similarity measures: $SAD$, $dSX$ and $dCEN$ are subjected to one-dimensional median filtering (widow size 11 samples) (module *MEDIAN 11*) and thresholded (module *TH*). The binary results are combined with the *AND* operator and subjected to binary one-dimensional median filtering (window size 11 samples) (module *MEDIAN BIN 11*). Finally, a binary variable containing information about vehicle presence is obtained.

Sample TSI images are presented in Figure 4. The binarization results of the three similarity measures are visualized as green (1) / blue (0) columns on the right side of the TSI. The red colour indicates fragments without car detection (i.e. carriageway). The delay introduced by the used median filtering is compensated later by extracting shifted fragments (on the presented images a 10-15 pixel "down" shift is necessary).

It is worth noting that in the TSI in Fig. 4b (at the bottom) a part of the dark car is erroneously detected as a carriageway. On the other hand, in TSI in Fig.4c (at the top) the distance between two cars turned out to be too small, and they could not be separated properly at this stage of the algorithm.

## B. *Patch Analysis*

Based on the detection results for a number of test sequences it has been found that in most cases the method described above can correctly extract individual vehicles. However, also situations when a patch contains more than one car or no car at all (in case of shadows) have been noticed. Furthermore, a slight difference in brightness or colour between the vehicle and the road, as well as occlusions cause serious classification problems. The later issue can be partially eliminated by appropriate positioning of the camera – directly over the road and at a fairly large angle. Sample patches, with erroneous detections, are presented in Figure 5.

Algorithms that allow to determine the number of vehicles in a given patch were proposed to improve the overall accuracy of the system. The solution can handle cases presented in Figure 5a-d. The other two situations should be considered as difficult and their correct classification requires further research.

Due to a completely different specificity, separate analysis procedures for day and night-time were developed.

*1) Analysis During Day-time:* The proposed method consist of the following steps:

- low-pass filtering (Gaussian) – distortion elimination and image smoothing.
- histogram stretching – a procedure similar to the available in Matlab software was used. There 1% of the brightest and darkest pixels is saturated respectively at the maximum and minimum values (0,255).
- another low-pass filtering (Gaussian) – the histogram stretching operation emphasizes noise present in the image and it should be remove prior further image analysis.
- calculation of the horizontal Sobel gradient – information about horizontal edges is used to determine the presence of an object and to eliminate shadows.

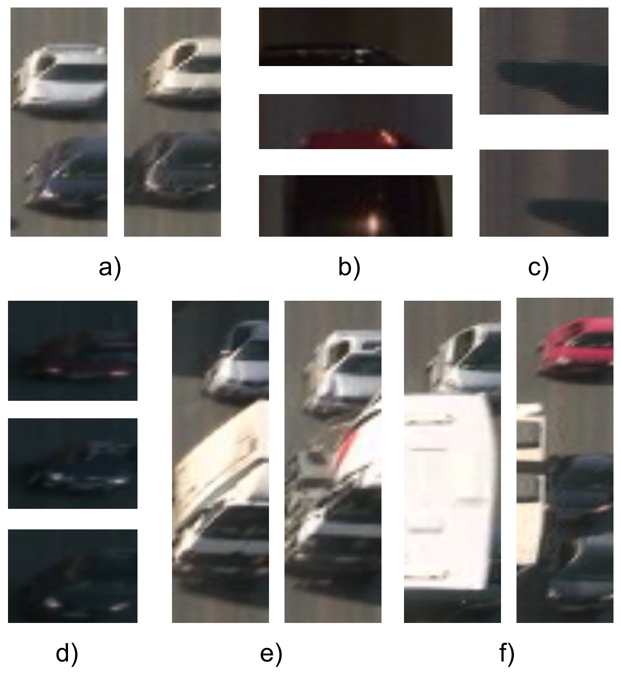

Fig. 5. Patches requiring further analysis. a) more than one vehicle in the patch, b) incorrectly detected patch (during night-time), c) detected shadow (cast by a car on a neighbouring line), d) very low contrast between vehicle and the background, e) vehicle partially occluded by a larger one, f) a vehicle on adjacent line causes distortions (in the left image it completely obscures the adjacent lane)

The gradient is calculated separately for each RGB colour component, then the result is thresholded and combined with an OR operator.

- shadow detection – to detect situations similar to the shown in Figure 5c a simple method was proposed that analyses edges and shadow areas (as shadow are regarded pixels with brightness below a certain threshold). The approach is illustrated in Figure 6.
  In the first step areas with horizontal edges and shadow pixels are determined. Then the bounding box around the edge area is computed. This allows, in most cases, to divide the patch into two parts: one containing the object (edges) and second with the background (no edges). For these parts greyscale histograms aggregated to 64 values are calculated. Then, the histograms are subtracted from each other, and, in addition, the histogram part corresponding to shadow areas (below a given threshold) is removed. Finally, values present in the obtained histogram are summed and then normalized by the size of the detection window. For the case presented in Figure 6 the resulting value is quite low (0.0827). If a vehicle is present in the patch, than on the resulting histogram the value corresponding to its brightens will be visible, thus the coefficient will be high.
  In addition, two factors are calculated: the ratio of number of pixels marked as edges to number of pixels marked as a shadow, and the ratio of number of pixels marked as edges and shadow to the bounding box area around the edges. For the patch presented in Figure 6 these values are respectively: 0.74 and 1.29. It was assumed that for shaded areas the first coefficient should be smaller than 2, i.e. edge and shadow areas should be approximate similar size (for vehicles the edge area is usually greater than the area of shadow) and the second coefficient should be greater than 0.6 i.e. the bounding box area is mostly ”filled” with points belonging to edges and shadows. To classify a given patch as containing a shadow all three conditions should be fulfilled.
  It is worth noting that the above conditions are also met by very dark vehicles, especially in low light conditions (cloudy day, rain). One possible solution could be the implementation of a deep shadow detection procedure – for example by analysing the shadow cast by a permanent element of the scene.

- analysis of patches with multiple vehicles – in the proposed method it is assumed that the carriageway is visible between vehicles (see Figure 4a). The approach is based on region growing segmentation performed on horizontal lines, separately for the left and right side, starting from the patch boarder. The threshold required to determine if the current pixel is similar to the previous ones is determined adaptively as 0.25 $\times$ current pixel value and 0.75 $\times$ previous threshold value. The segmentation is performed in RGB colour space. The resulting masks (for the left and right part) are subject to morphological dilation and then combined by the AND operator. In this way, portions of the carriageway are detected. In the final step the number of separate vehicles is determined. It should be noted that this analysis is performed only for patches with a height greater than a specified threshold. The method is illustrated in Figure 7.
  During preliminary research some other approaches to this issue were evaluated. First, the possibility of using information about colour to determine the number of vehicles present on a patch was tested. Experiments in HSV, YCbCr and CIE Lab colour spaces were performed. Unfortunately, analysis of the obtained histograms revealed that this approach can work only for vehicles with a very clear difference in colours (e.g. red and green). In other cases, it was impossible, even ”manually” to point out at the histogram the maximum corresponding to a given vehicle. The probable cause is the fairly inhomogeneous lighting, including reflections and shadows.
  Second, local background modelling for a VDL was evaluated. The information could be very useful in vehicle segmentation on a given patch. However, problems similar to those occurring when modelling the background for the whole scene were noted – mainly with distortions caused by shadows.
  The development of methods able to handle the difficult cases – Figure 4e and f will be part of future research.

- estimation the final number of vehicles – the analysis described above is supplemented by counting the overall number of horizontal edges present in the patch. If the number is less than a given threshold, the patch is regarded as a false detection (i.e. without a vehicle).

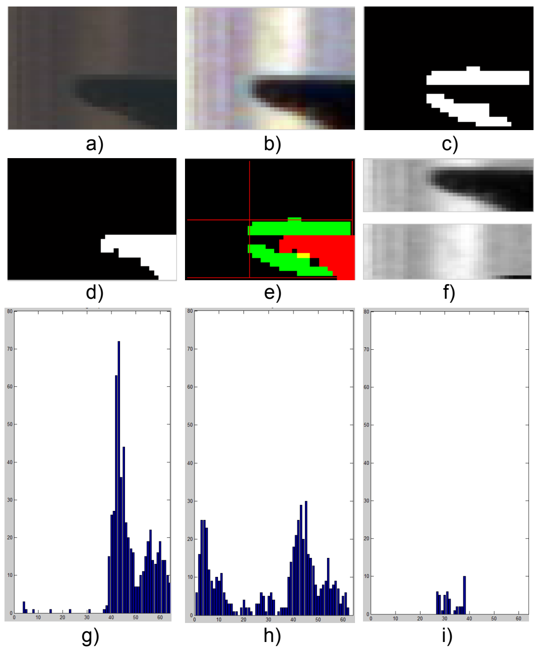

Fig. 6. Sample shadow detection. a) input image, b) image after histogram stretching, c) horizontal edges, d) shadow areas, e) combination of images c) and d) with bounding box around the edges, f) part with the edge area (top) and without edges (bottom), g) histogram for the non-edge area, h) histogram for the edge area, i) difference between histograms g) and h) with removed values corresponding to shadow areas.

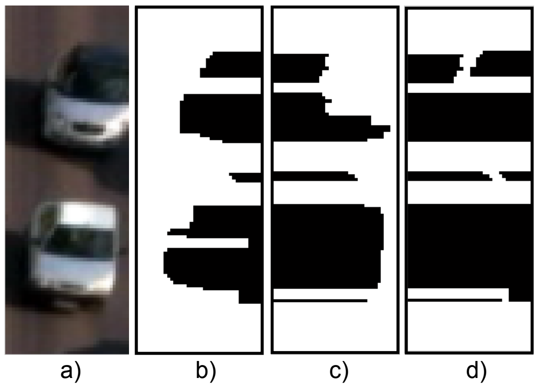

Fig. 7. Sample multiple vehicles detection. a) input image, b) and c) region growing segmentation results, the procedure started for left and right edge of the patch (additional morphological dilation was performed), d) combination of images b) and c) – AND operator.

*2) Analysis During Night-time:* The basis of the patch analysis method during nigh-time is the detection of vehicle headlights. First, binarization of the gryscale image with a quite high threshold (200) is performed, followed by a single-pass connected component labelling. The area and centroids of detected object are computed. Finally, the presence of two objects with similar vertical and different horizontal coordinates is determined.

### C. Algorithm Evaluation

The algorithm has been implemented in C++ using the OpenCV library and evaluated on a number of test sequences registered by a camera located above a busy intersection on one of the main street in Krakow. Sequences were recorded under different conditions: sunny day, cloudy, day, rainy day and night-time. In total over 53000 frames were analysed (about 30 minutes). It is also worth to mention, that all the tests were performed on the same set of parameters i.e. they were not fine-tuned to particular sequences or even weather conditions.

TABLE II. PROJECT RESOURCE UTILISATION

| Resource | Used | Available | Percentage |
|---|---|---|---|
| FF | 8759 | 106400 | 8 % |
| LUT 6 | 6785 | 53200 | 12 % |
| SLICE | 2935 | 13300 | 22 % |
| BRAM_18 | 4 | 280 | 1 % |
| BRAM_36 | 33 | 140 | 23 % |
| DSP 48 | 3 | 220 | 1 % |

In the current version of the system only one virtual detection line placed in front of the traffic lights was used. During evaluation the total number of detections (actual and returned by the algorithm), as well as analysis results of particular patches were counted. In the second case, the typical binary classification measures were used:

- TP – true positive – correctly detected vehicle,
- TN – true negative – correctly detected road or distortion,
- FP – false positive – road detected as vehicle,
- FN – false negative – vehicle detected as road.

The obtained results are presented in Table I The mean accuracy of the proposed system was 96%.

The main cause for false positives were vehicles stopped at the VDL, which were sometimes counted more than one time. Almost all false negatives were caused by black cars misclassified as shadow. Therefore, the day-time procedure certainly requires some refinement. In case of sequences registered at night-time, all the missed detection were the result of low brightness of the headlights.

It is worth to mention, that the method allowed to eliminate many potential false detection, which is indicated by the quite high number of true negatives.

## IV. Heterogeneous Vision System

The algorithm described in the previous section, has been implemented on the Zynq device (XC7Z020 CLG484 -1 AP SoC) available at the ZC702 evaluation board made by Xilinx. The video stream, transmitted in HDMI standard, was supplied to the system via the AES-FMC-DVI-G FMC (FPGA Mezzanine Card) module. During implementation the ISE, EDK and ISim software from Xilinx was used. Scheme of the proposed system is presented in Figure 8. FPGA resource usage for the hardware part of the system is summarized in Table II.

The algorithm has been divided into hardware (VDL) and software part (vehicle patch analysis) described respectively in Subsections III-A and III-B. The data transfer between these two parts of the system is handled by the AXI bus. Particularly, the pixels from the VDL and values of three similarity measures $SAD$, $dSX$ and $dCEN$ are transmitted.

TABLE I. EVALUATION RESULTS OF THE PROPOSED VEHICLE DETECTION AND COUNTING ALGORITHM

| Sequence | Cars – actual | Cars – detected | No. of patches | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|
| Sunny | 82 | 83 | 96 | 76 | 2 | 11 | 7 |
| Cloudy | 179 | 166 | 210 | 163 | 28 | 3 | 16 |
| Rainy | 123 | 116 | 151 | 116 | 28 | 0 | 7 |
| Night-time | 93 | 88 | 116 | 88 | 23 | 0 | 5 |
| Overall | 627 | 601 | 765 | 587 | 114 | 22 | 42 |

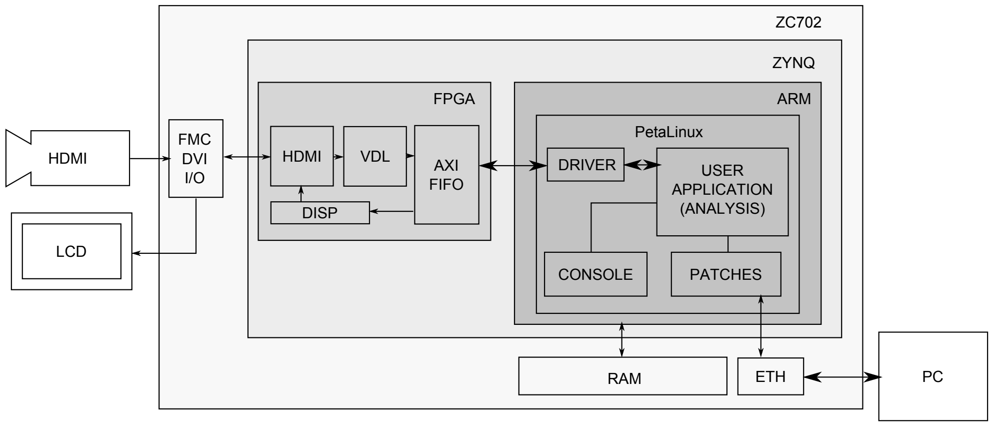

Fig. 8. Scheme of the proposed hardware-software system

## A. Hardware Part

During designing the hardware part it has been assumed that the target number of VDLs will be greater than one. Therefore all operations that are common to the modules (colour space conversion from RGB to greyscale, Sobel edge detection, Gaussian filtering, Census transform) were performed for the entire video frame. Preprocessed data are fetched into a single VDL module, whose scheme is analogous to those presented in Figure 3.

The required one dimensional median filtering was realised using the Bitonic Merge Sort algorithm. The VDL module has a local image buffer (Block RAM memory – one 36K module), which stores only necessary data from the previous frame ($N-1$). This solution allows for a significant memory saving, compared with storing the entire frame, which would have to be implemented in external RAM.

All modules were implemented in Verilog hardware description language. The hardware modules results were compared with those obtained from the software application (C++) in the ISim simulation tool. Scheme of the proposed module is presented in Figure 9. In addition to the previously mentioned modules, the scheme includes the *POSITION* module, which is responsible for determining the current position in the image using video synchronization signals (*de, hsync, vsync*) and information about image resolution. A single VDL transmits to the software part RGB pixels values (after Gaussian low-pass filtering), as well as computed similarity measures and detection results.

Additional elements of the hardware system, presented in Figure 8, are: *AXI FIFO* buffers used in the data transfer between *FPGA* and the processor system and HDMI – communication with the FMC module, responsible for video stream receiving and displaying.

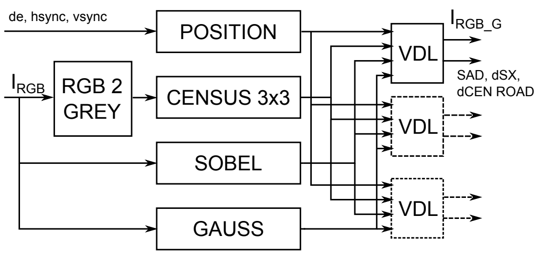

Fig. 9. Scheme of the VDL module

## B. Software Part

In the software part the PetaLinux operating system is used. The communication with the hardware part is realised with an AXI bus driver (*DRIVER*). The analysis algorithm, described in Subsection III-B, is implemented as a user application. It has been divided into three parts:

- patch creation using data obtained from the driver, patch saving,
- analysis in day-time conditions,
- analysis in nigh-time conditions.

The user application displays informations about detections, as well as other diagnostic data on a console. Additionally, it is possible to connect to the system using Secure FTP protocol and download the obtained vehicle patches. Tests showed, that the communication between the elements of the system works correctly, and the ARM processor is efficient enough to analyse patches in real-time using the proposed algorithms.

## V. FUTURE WORK

The described system is a prototype that can be further developed. First of all, it seems necessary to propose better algorithms for the analysis of patches to improve the overall accuracy. Proper identification of large vehicles (buses, trucks) makes serious difficulties in the current version of the algorithm. It is also possible to expand the functionality: introduce vehicle classification, speed estimation, determination of the length of the queue and possibly unusual event detection. It is worth noting that not all this features must rely on the use of VDLs. Some may require analysis at the level of the whole image. The parallel implementation of the various parts of the algorithm is easier through the use of the reconfigurable resources for image processing. Implementation of many VDL – for example, three on each lane – should allow to increase reliability. The proposed algorithm division between hardware and software is not final. Moving further elements to FPGA resources (e.g. local histogram stretching) seems to be a very promising direction of work. It is also worth to carefully examine the parameters and thresholds used in the algorithm and make an attempt to optimize them.

## VI. Conclusion

The article demonstrates the usefulness of heterogeneous Zynq SoC to build a system for a smart camera dedicated to traffic monitoring. To achieve this, the following issues have been resolved: hardware-software architecture required to control the acquisition of the HDMI video signal to the system, the AXI bus based communication between the FPGA and ARM processor, AXI bus driver for the PetaLinux system and configuration of the operating system itself. An algorithm for detecting and counting vehicles was also proposed, that allowed to demonstrate the capabilities of the developed architecture. The accuracy of the algorithm reached 96%.

Implementation of the algorithm was done partially in hardware (FPGA) and software (ARM with PetaLinux) and was positively verified on the ZC702 platform from Xilinx. The system allows to process 50 frames with a resolution of $720 \times 576$ pixels per second. The obtained results indicate that Zynq SoC provide a good basis for the implementation of advanced video algorithms and building smart cameras as they combine the advantages of reconfigurable circuits and general purpose processor system.

## Acknowledgment

The work presented in this paper was supported by SPIN Project co-funded by the European Union under European Social Fund (ESF) within Human Capital Operational Programme, Priority VIII: Regional human resources for the economy, Sub-measure 8.2.1 Support to cooperation of scientific environment and enterprises (first and second author) and the AGH Univeristy of Science and Technology project no. 11.11.120.612 (third author)

## References

[1] M.-T. Yang, R.-K. Jhang, and J.-S. Hou, “Traffic flow estimation and vehicle-type classification using vision-based spatial-temporal profile analysis,” *Computer Vision, IET*, vol. 7, no. 5, pp. 394–404, October 2013.

[2] J.-W. Hsieh, L.-C. Chen, D.-Y. Chen, and S.-C. Cheng, “Vehicle make and model recognition using symmetrical SURF,” in *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, Aug 2013, pp. 472–477.

[3] A. N. Belbachir, *Smart cameras*, A. N. Belbachir, Ed. Springer, 2010.

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

[5] R. Dobai and L. Sekanina, “Image filter evolution on the Xilinx Zynq Platform,” in *Adaptive Hardware and Systems (AHS), 2013 NASA/ESA Conference on*, June 2013, pp. 164–171.

[6] S. Gilliland, P. Govindan, T. Gonnot, and J. Saniie, “Performance evaluation of FPGA based embedded arm processor for ultrasonic imaging,” in *Ultrasonics Symposium (IUS), 2013 IEEE International*, July 2013, pp. 519–522.

[7] J. Monson, M. Wirthlin, and B. Hutchings, “Implementing high-performance, low-power FPGA-based optical flow accelerators in C,” in *Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on*, June 2013, pp. 363–369.

[8] M. Russell and S. Fischaber, “OpenCV based road sign recognition on Zynq,” in *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, July 2013, pp. 596–601.

[9] J. Cao and L. Li, “Vehicle objects detection of video images based on gray-scale characteristics,” in *Education Technology and Computer Science, 2009. ETCS ’09. First International Workshop on*, vol. 2, March 2009, pp. 936–940.

[10] A. Ghasemi and R. Safabakhsh, “A real-time multiple vehicle classification and tracking system with occlusion handling,” in *Intelligent Computer Communication and Processing (ICCP), 2012 IEEE International Conference on*, Aug 2012, pp. 109–115.

[11] A. Glowacz, Z. Mikrut, and P. Pawlik, “Video detection algorithm using an optical flow calculation method,” in *Multimedia Communications, Services and Security*, ser. Communications in Computer and Information Science, A. Dziech and A. Czyzewski, Eds. Springer Berlin Heidelberg, 2012, vol. 287, pp. 118–129.

[12] H. Yalcin, M. J. Black, R. Collins, and M. Hebert, “A flow-based approach to vehicle detection and background mosaicking in airborne video,” Proc. Computer Vision and Pattern Recognition, Tech. Rep., 2005.

[13] C. Pang, W. Lam, and N. H. C. Yung, “A method for vehicle count in the presence of multiple-vehicle occlusions in traffic images,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 3, pp. 441–459, Sept 2007.

[14] N. Mithun, N. Rashid, and S. Rahman, “Detection and classification of vehicles from video using multiple time-spatial images,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 3, pp. 1215–1225, Sept 2012.

[15] M. Gorgon, P. Pawlik, M. Jablonski, and J. Przybylo, “FPGA-based road traffic videodetector,” in *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, Aug 2007, pp. 412–419.

[16] J. Jin, V. D. Nguyen, S. J. Lee, and J.-W. Jeon, “FPGA design and implementation of a real-time vehicle detection system,” in *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, Oct 2012, pp. 204–207.

[17] S. Szczepanski, W. M., B. Pankiewicz, M. Klosowski, and R. Zaglewski, “FPGA and ASIC implementation of the algorithm for trac monitoring in urban areas,” *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 2011.

[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, no. 1, pp. 1–18, 2012.

[19] M. Wojcikowski, R. Zaglewski, B. Pankiewicz, M. Kosowski, and S. Szczepanski, “Hardware-Software Implementation of a Sensor Network for City Traffic Monitoring Using the FPGA- and ASIC-Based Sensor Nodes,” *Journal of Signal Processing Systems*, vol. 71, no. 1, pp. 57–73, 2013.

[20] F. Pletzer, R. Tusch, L. Boszormenyi, and B. Rinner, “Robust traffic state estimation on smart cameras,” in *Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on*, Sept 2012, pp. 434–439.