Real-Time Implementation of Background Modelling Algorithms in FPGA Devices

Tomasz Kryjak(✉) and Marek Gorgon

AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Krakow, Poland
{tomasz.kryjak,mago}@agh.edu.pl

Abstract. The article discusses the possibilities of hardware implementation of foreground object segmentation and background modelling algorithms in FPGA. The potential benefits, as well as challenges and problems associated with porting algorithms from general-purpose processors (CPU) to reconfigurable logic (FPGA) are presented. Also several hardware implementation of well known method are reviewed: GMM, Codebook, Clustering, ViBE and PBAS.The last algorithm was also evaluated on the SBI dataset.

Keywords: Real-time image processing · Embedded systems · Smart cameras · Background modelling · Foreground object segmentation

1 Introduction

Foreground object segmentation and background modelling is one of the fundamental steps in a number of automatic video stream analysis system. This involves human detection, tracking and action recognition, abandoned or stolen object detection (advanced video surveillance systems) and vehicle traffic monitoring (video based ITS). The method is conceptually very simple. Objects are detected by comparing the current frame and the background representation (background model). Sometimes, the algorithms are supported by moving object detection e.g. using optical flow.

However, due to several factors: camera jitter, variable lighting conditions (slow or rapid changes), objects to background similarity, movement of background object (e.g. flowing water), stopped objects (pedestrian waiting for green traffic light) and objects which started to move (a car left a parking lot) the issue is not easy. A very good and extensive review of most common approach can be found in the work by Thierry Bouwmans [4].

Unfortunately, usually the methods which allow to obtain object mask of better quality are also computationally complex. Furthermore, segmentation is only one of the first stages of a vision system. In recent years, the dynamic development of so-called smart cameras can be observed. In this solution, the image processing, analysis and recognition is performed immediately after acquisition. This allows for the computation decentralization and avoid image quality deterioration due to compression. Such solutions are offered by leading manufacturers of video surveillance cameras Bosch, Ganz, Pelco.

V. Murino et al. (Eds.): ICIAP 2015 Workshops, LNCS 9281, pp. 519–526, 2015.
DOI: 10.1007/978-3-319-23222-5_63

1.1 FPGA Devices

When designing a smart camera, parameters like energy efficiency (GOPS/W, GFLOPS/W), as well as computation parallelisation and acceleration are very important. In systems that require image processing in real time and high quality foreground obejct masks the GPP implementation may not meet expectations due to insufficient performance computing. Therefore, FPGA (Field Programmable Gate Arrays) are a very interesting computing platform for these kind of applications. FPGA are build of many simple logic elements (flip-flops, LUTs and multiplexers). They also include internal memory resources, DSP modules and advanced I/O capabilities (with support for different standards). What's more, the devices can be reconfigured (by downloading a bitstream file) many times, even after installation in the target system. The FPGAs are used in all kinds of vision systems (introduction in the book by Bailey [2]), wire and wireless telecommunication networks, as well as automotive, defence and space industries.

FPGAs, unlike other platforms on which parallel implementation of algorithms is possible e.g. multi-core CPUs, DSPs or GPUs, provide an almost full flexibility when designing the system architecture. Moreover, the filed reprogrammability allows to preform algorithms update (firmware updates). This is not possible in ASIC (Application Specific Integrated Circuit) or ASSP (Application Specific Standard Product) based solutions. It is also worth to mention, that in recent years heterogeneous SoC (System on Chip) devices have been proposed. For example, the Zynq from Xilinx combines reconfigurable resources and an efficient dual-core ARM processor. This allows for easy hardware-software co-design.

1.2 FPGA Logic Design Methodology

The first stage always involves the analysis of the computational task, as well as an attempt to propose a parallelisation. In the fine-grain FPGA logic all types of parallelism according to Flynn's classification (SISD, SIMD, MISD, MIMD) can by realised. In the second step, the so-called software model is implemented. This is an software application, which performs calculations in the same way as the hardware (i.e. bit accurate model). At this point it is worth to discuss some limitations of FPGAs. First of all, fixed-point arithmetic is preferred[1]. Secondly, the typical operating frequency reaches 200-300 MHz. This is enough for VGA/PAL and HD processing, where the so-called pixel clock is respectively 25-27 and 150 MHz, but almost only when data is processed in pipeline. A serious challenge are algorithms, which require image content dependent data access (e.g. region growing segmentation) or storage of large amount of data (need for external RAM access).

For logic design the so-called hardware description languages (HDL) are used. The two most common are Verilog and VHDL. They provide full control of the

[1] In recently presented FPGAs from Altera hardware support for single precision floating arithmetic is provided.

created system. Other solutions involve high level synthesis (HLS) tools, which allows C/C++ to HDL code conversion and extensions to packages like Matlab (HDL Coder, System Generator) or LabView.

The designed system is tested in two stages. First simulations are performed. The aim is to achieve full compliance with the described earlier software model. Secondly, the system is evaluated in hardware i.e. on an FPGA development board with video stream input/output.

At this stage, several additional issues not directly related to the algorithm implementation, but important for the functionality of the entire video system should be considered. Firstly, it is necessary to communicate with the video stream source. For prototyping a good solution is HDMI, which is very intuitive and easy to use.

Second, despite the significant increase of FPGA memory resource, their size is too small to store a full image frame in STD resolution. Therefore, larger chunks of data i.e. background model or previous frame should be saved in external memory (usually DDR). These have large capacities, but the maximum data transfer rate is limited.

The third element is the broadly understood results visualisation and transfer. In the simplest case, they are displayed on an LCD screen. However, it possible to send them in the form of meta-data with a compressed video stream.

2 Background Modelling Implemented in FPGA

With the development od FPGAs, they became an interesting platform for vision systems implementation. The firsts designs were very simple image processors like: histogram computation, convolutional or median filters or colourspace conversions. Nowadays, it is possible to carry out complex operation like object classification (HOG+SVM, Haar + AdaBoost) or tracking.

The first works related to background modelling date back to year 2005. In the article by Appiah et al. [1] an FPGA implementation of a Gaussian Mixture Models (GMM) based method was descried. For each pixel, K background variants were stored, each composed of a mean and a weight. In Gorgon et al. [7] an approach based on average brightness calculation and periodic re-scaling mechanism was proposed. Real-time processing for a 768×576 pixels @ 25 fps video stream was reported.

Most publications on this subject appeared in the last four years. Definitely worth mention are:

– Hardware implementation of the GMM algorithm in Genovese and Napoli [6] – real-time image processing of 1280×720 pixels @ 20 fps video stream.
– FPGA implementation of Horprasert method in Gomez et al. [15] – real-time image processing of 1024×1024 @ 32.8 fps video stream.
– FPGA implementation of Codebook method in Gomez et al. [16] – real-time image processing of 768×576 @ 60 fps video stream.

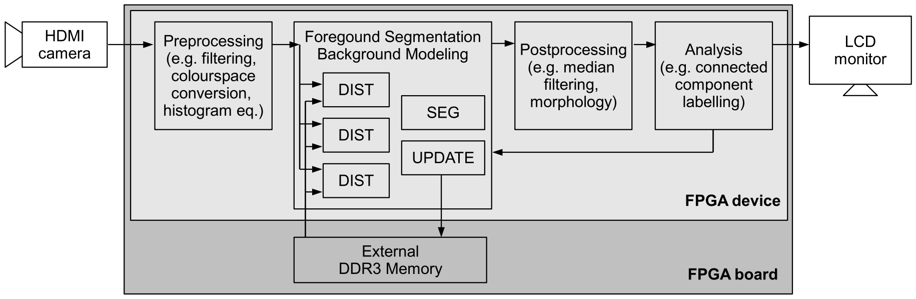
HDMI camera
Preprocessing (e.g. filtering, colourspace conversion, histogram eq.)
Foregound Segmentation Background Modeling
DIST
DIST
DIST
SEG
UPDATE
Postprocessing (e.g. median filtering, morphology)
Analysis (e.g. connected component labelling)
LCD monitor
FPGA device
External DDR3 Memory
FPGA board

Fig. 1. Scheme of an FPGA-based system for background modelling and foreground segmentation.

2.1 The General Scheme

FPGA implementation of background modelling methods involves several issues. Firstly, most of the method that perform well in the rankings accepted by the community (e.g. *changedetection.net*) are multivariate i.e. the background model for a given pixel has more than one value. Examples are: GMM, Clustering, Codebook, ViBE, PBAS, as well as recent proposals like FTSG (Flux Tensor with Split Gaussian models). Such solutions are quite straightforward to parallelise, as shown in Fig. 1 – modules DIST.

Secondly, as the fixed point arithmetic is preferred in FPGA, the research should involve representation impact analysis (i.g. a comparison with the floating point version). Thirdly, modern FPGAs do not have enough internal memory resources to store a complex background model. For a 'typical" algorithm, for each pixel from the video stream, the corresponding model should be provided and after update saved back in RAM. On the VC707 platform from Xilinx (used by the Authors of this paper) for a video stream 720× 576 @ 50 fps the maximal transfer rate is 2048 bits per pixel clock period (at 27 MHz). For such algorithms as GMM this is satisfactory, but for example in case of ViBE or PBAS it forces some limitations.

2.2 Implemented Algorithms

We designed hardware modules of three foreground object segmentation algorithms: Clustering [10], ViBE [9], [11] and PBAS [13], [12].

Clustering. The Clustering algorithm was described in Butler et al. [5] and is a simplification of GMM. The background model for a single pixel consists of K variants, each formed by a mean and weight. The lack of the standard deviation or covariance results in a compact representation and simpler calculations.

The main objective of the research described in Kryjak et al. [10] was to show that in a modern FPGA it is possible to implement a complex background modelling algorithm and obtain real-time processing for a 1920 × 1080 @ 60 fps video stream [2]. Several improvements and modification were proposed to the algorithm, among others the use of CIE Lab colourspace and the use of edges in the background model. Furthermore, during segmentation difference in brightness, colour and texture (Normalised Gradient Difference) was used. The module was evaluated on the Wallflower dataset. The results were comparable to the state of the art (in year 2012). The vision system was implemented on the Xilinx ML605 FPGA board with Virtex 6 device. The model for a single pixel had 141 bits (3 variants, in each CIE Lab pixel (32 bits), edge information (9 bits) and weight (6 bits)) and the bandwidth to external RAM was 33459 Mb/s (HD resolution).

ViBE. The ViBE (Visual Background Extractor) algorithm was proposed in 2009 by O. Barnich and M. Van Droogenbroeck [3]. It is a hybrid method based on sample buffer (non recursive) and recursive approach. The background model consists of $N = 20$ pixel samples (greyscale or colour). The foreground - background classification is based on calculating the distance (e.g. Euclidean) between the current pixel and all samples. If at least $\#_{min}$ are smaller than a given threshold, than the pixel is considered as background.

The other features of the algorithm are:

- background model initialization based on a single frame – the N samples buffer is filled with values randomly selected from a 3×3 context,
- conservative update policy (updated are only pixels regarded as background),
- random factors in the update process (the update id performed with some probability, the sample to modify is selected randomly),
- neighbourhood update procedure, in order to eliminate “ghost” classification errors.

Hardware implementation of the ViBE algorithm on the ML 605 platform is described in Kryjak et al. [9]. The general scheme is similar to that presented in Fig. 1. Real-time processing of 640×480 @ 60 fps video stream was obtained. The model for a single pixel had 460 bits (20×23bits) and the bandwidth to external RAM was 13645 Mb/s.

An extended version of this module was described in Kryjak et al. [11]. There, the algorithm was adapted to a moving camera scenario. Camera displacement between two consecutive frames estimation was implemented. It was based on 3×3, SAD, block matching approach. The displacement was not computed for each pixel (i.e. dense optical flow), but only for certain points selected in 32×32 blocks using Harris corner detection. Then, the final displacement was defined as the median values from particular blocks. The system was implemented on the VC707 board with Virtex 7 FPGA from Xilinx. Real-time image processing was obtained for a $720\times$ 576 @ 50 fps video stream. It is worth noting, the solution is characterized by only 4 W power dissipation.

[2] When the module was designed, i.e. in 2012, no other hardware implementation able to process HD video stream was reported.

PBAS. The PBAS (Pixel-Based Adaptive Segmenter) described in Hofmann et al. [8] is based on ViBE. Two additional parameters were added: R (variable similarity threshold) and T (variable update rate). Both are updated using properties of the model for a given location. This improves the segmentation performance, but increases the memory requirements (additional for each pixel location R, T and some auxiliary data has to be stored).

FPGA implementation of the PBAS algorithm was described in Kryjak et al. [13]. Because of limited transfer rate to external RAM, the number of samples had to be reduced from 35 proposed by the authors of the algorithm to 19. This resulted in slight deterioration of segmentation performance (evaluation on the *changedetection.net* dataset). Finally, real-time image processing for a 720×576 @ 50 fps video stream was obtained. It is also worth to emphasize, that the model for a single pixel had the size of 1008 bits ($3 \times (19 \times 16\text{bits} + 2 \times 16\text{bits})$ – 19 samples containing 8-bit colour component and 8-bit minimum distance value, parameters R and T (16 bits), all these for three components). Therefore, the bandwidth to external RAM was 39867 Mb/s. The measured power dissipation was less than 7 W.

In Kryjak et al. [12] the PBAS algorithm was supported with feedback from object analysis module. It was designed to improve segmentation results is scenarios, where it is necessary to distinguish between real objects, which have stopped (e.g. abandoned luggage) and so-called "ghosts". During experiments the *Intermittent Object Motion* dataset from *changedetection.net* was used. Two features were utilized: motion (determined by consecutive frames differencing) and edges (computed for the foreground, background and object mask). The first was used to determine whether the object is not moving. The second, to distinguish between stooped objects (mask edges are similar to current frame edges) and ghost (mask edges are similar to background model edges). It is worth noting, that this rather simple approach requires connected component labelling i.e. transition from individual pixel analysis to objects (connected pixel group) analysis. In a software implementation this step is quite straightforward, however in a pipeline vision systems it is quite challenging. The system was implemented on the VC707 platform with Virtex 7 FPGA. Real-time image processing for a 720 × 576 @ 50 fps video stream was obtained.

2.3 Evaluation on the SBI Dataset

The SBI (Scene Background Initialization) dataset [14] was prepared for the Scene Background Modeling and Initialization Workshop. In consists of seven video sequences and 7 reference backgrounds. In addition 8 metrics are proposed to assess the initialization process. Detailed information is available on the http://sbmi2015.na.icar.cnr.it website.

After initial analysis of the sequences, from the 3 methods: Clustering, ViBE and PBAS, the last was chosen for evaluation mainly due to the implemented feedback from the object analysis module. However, it should be noted that PBAS is not a "typical" background initialisation method. The initial model is obtained using the first frame from the sequence, which for the rather short

Table 1. Result of the PBAS FPGA implementaion on the SBI dataset

Sequence	AGE	EPs	pEPs	CEPs	pCEPS	MSSSIM	PSNR	CQM
HallAndMonitor	4.5412	1962	0.0232	11	0.0001	0.9874	24.9338	31.5584
HighwayI	10.3774	10596	0.1380	7614	0.0991	0.7204	20.7564	28.1407
HighwayII	4.7919	2976	0.0388	440	0.0057	0.9490	24.6061	30.7008
CaVignal	15.1816	1682	0.0618	19	0.0007	0.9301	18.6021	27.6635
Foliage	21.3365	7063	0.2452	2676	0.0929	0.8328	17.6953	25.8441
PeopleAndFoliage	19.1997	15779	0.2055	4990	0.0650	0.8848	19.0184	26.1430
Snellen	62.3326	16310	0.7866	14306	0.6899	0.3721	10.8639	22.5839

test sequences has major impact on the experiment. Moreover, no explicit representation of the background model is provided. It consists of randomly located samples. Thus, to create a model in the form of an "image" for example the median value from the buffer should be selected. Furthermore, the calculations are independent for RGB components, which result in small errors in the background image. The obtained results are summarized in the Tab. 1.

For two sequences *CaVignal* and *HallAndMonitor* the obtained results are quite correct. The used edge analysis approach allowed to eliminate ghosts. For *HighwayII* the results are partially correct. In regions, where the movement is not very intensive, the ghosts were eliminated. However, some errors due to the median based background image "extraction" occur.

For other sequences, due to constantly moving objects (foliage or cars) it is impossible to eliminate the ghosts with the implemented approach. This issue should be further investigated.

3 Summary

In this paper examples of FPGA based hardware implementations of foreground segmentation and background modelling algorithms were presented. The general concept and design approach, as well as three modules: Clustering, ViBE and PBAS were discussed. The last was also evaluated on the SBI dataset. Using FPGA devices it is possible to obtain real-time image processing for VGA/PAL or even HD video stream resolution. The proposed modules could be used in low-power smart cameras.

Acknowledgments. The work presented in this paper was supported by AGH University of Science and Technology projects number 15.11.120.476 (first author) and 11.11.120.612 (second author).

References

1. Appiah, K., Hunter, A.: A single-chip FPGA implementation of real-time adaptive background model. In: Proceedings of the IEEE International Conference on Field-Programmable Technology, pp. 95–102 (2005)
2. Bailey, D.G.: Design for embedded image processing on FPGAs, John Wiley & Sons (2011)

3. Barnich, O., Van Droogenbroeck, M.: ViBE: A Universal Background Subtraction Algorithm for Video Sequences. IEEE Transactions on Image Processing **20**(6), 1709–1724 (2011)
4. Bouwmans, T.: Traditional and recent approaches in background modeling for foreground detection: An overview. Computer Science Review **1112**, 31–66 (2014)
5. Butler, D., Sridharan, S., Bove Jr, V.M.: Real-time adaptive background segmentation. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 349–352 (2003)
6. Genovese, M., Napoli, E.: FPGA-based architecture for real time segmentation and denoising of HD video. Journal of Real-Time Image Processing **8**(4), 389–401 (2013)
7. Gorgon, M., Pawlik, P., Jablonski, M., Przybylo, J.: FPGA-based road traffic videodetector. In: 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, pp. 412–419 (2007)
8. Hofmann, M., Tiefenbacher, P., Rigoll, G.: Background segmentation with feedback: the pixel-based adaptive segmenter. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 38–43 (2012)
9. Kryjak, T., Gorgon, M.: Real-time implementation of the ViBe foreground object segmentation algorithm. In: Federated Conference on Computer Science and Information Systems (FedCSIS), pp 591–596 (2013)
10. Kryjak, T., Komorkiewicz, M., Gorgon, M.: Real-time background generation and foreground object segmentation for high defnition colour video stream in FPGA device. Journal of Real-Time Image Processing **9**(1), 61–77 (2014)
11. Kryjak, T., Gorgon, M.: Real-time implementation of foreground object detection from a moving camera using ViBE algorithm. Computer Science and Information Systems **11**(4), 1617–1637 (2014)
12. Kryjak, T., Komorkiewicz, M., Gorgon, M.: Real-time foreground object detection combining the PBAS background modelling algorithm and feedback from scene analysis module. International Journal of Electronics and Telecommunications **60**(1), 61–72 (2014)
13. Kryjak, T., Komorkiewicz, M., Gorgon, M.: Hardware implementation of the PBAS foreground detection method in FPGA. In: Proceedings of the 20th International ConferenceMixed Design of Integrated Circuits and Systems (MIXDES), pp. 479–484 (2013)
14. Maddalena, L., Petrosino, A.: Towards Benchmarking Scene Background Initialization. arXiv:1506.04051 (publicly avaliable at http://arxiv.org/abs/1506.04051) (2015)
15. Rodriguez-Gomez, R., Fernandez-Sanchez, E.J., Diaz, J., Ros, E.: FPGA Implementation for Real-Time Background Subtraction Based on Horprasert Model. Sensors **12**(1), 585–611 (2012)
16. Rodriguez-Gomez, R., Fernandez-Sanchez, E.J., Diaz, J., Ros, E.: Codebook hardware implementation on FPGA for background subtraction. Journal of Real-Time Image Processing **10**(1), 43–57 (2015)