Implementation of Advanced Foreground Segmentation Algorithms GMM, ViBE and PBAS in FPGA and GPU – A Comparison

Bartłomiej Bulat, Tomasz Kryjak, and Marek Gorgon

AGH University of Science and Technology
Al. Mickiewicza 30, 30-059, Kraków, Poland
bartek.bulat@gmail.com, {kryjak,mago}@agh.edu.pl

Abstract. The article presents the results of implementing advanced foreground object segmentation algorithms: GMM (Gaussian Mixture Model), ViBE (Visual Background Extractor) and PBAS (Pixel-Based Adaptive Segmenter) on different hardware platforms: CPU, GPU and FPGA. The influence of the architecture on the segmentation accuracy and feasibility to perform real-time video stream processing was analysed. Also the limitations resulting from the specific features of GPU and FPGA were pointed out. Furthermore, the possible use of different platforms in advanced vision systems was discussed.

Keywords: image processing, image analysis, foregorund object segmentation, GMM, ViBE, PBAS, FPGA, GPU.

1 Introduction

Foreground object segmentation is a very important stage in digital video stream processing and analysis systems. It is based on determining and thresholding the differential image between the current frame and the background model. The term background model refers to a representation of the "empty scene" (i.e. without objects of interest). This approach is usually referred to as background subtraction and modelling. A comprehensive review of the most common algorithms is presented in [2].

The most popular method seems to be GMM (Gaussian Mixture Model), which was proposed in 1999 by Stauffer and Grimson [8]. It is constantly developed, modified and improved, and its implementations are available in OpenCV image processing library and Matlab software.

Interesting are also two fairly new proposals ViBE [1] and PBAS [5]. They are based on a similar principle and allow to obtain good segmentation results – according to the *changedetection.net 2012* ranking [4][1]. These three methods will be discussed in this paper.

It is worth to consider, whether working on acceleration of foreground objects segmentation is justified. It should be emphasised that the segmentation is

[1] http://www.changedetection.net/

only one of many elements, beside image pre-processing, connected component labelling, feature extraction, tracking and classification. While properly implemented, a segmentation method can operate in real-time for moderate resolutions on a standard PC. However, the whole vision system can not, particularly for higher resolutions. Thus, the acceleration of some computations in a GPU or FPGA allows to "relive" the CPU and the obtained computing power can be used in other stages of the vision system – usually recognition and classification.

In Section 2 the methods GMM, ViBE and PBAS are very briefly discussed. In Section 3 the used architectures are described. Then, in Section 4 the obtained results are presented and analysed. The article ends with a summary.

2 The Analysed Foreground Segmentation Methods

2.1 GMM

There are two different names of the algorithm [8] used in the literature: GMM (Gaussian Mixture Model) and MOG (Mixture of Gaussians). The background model for a given pixel consists of k Gaussian distributions described by triples: $(\omega_n, \mu_n, \Sigma_n)$, where ω_n – weight, μ_n – average value, Σ_n covariance matrix of the colour components (RGB model). The total number of distributions usually varies between 3-5.

Updating the model requires sorting the distributions, calculating the distances between distributions and the current pixel and an optional update of the distributions parameters. It is worth noting that the computations are independent for each pixel and no neighbourhood operations are involved.

The method was implemented in FPGA several times. In this paper, the results presented in the work [3] are used for comparison. GPU realisations are also described in the literature. In [9] several optimizations were discussed: the use of pinned memory, combining data transfers (memory coalescing), a special organization of the data structure and asynchronous kernels. The modifications made it possible to process more than 30 frames per second, even for HD resolution.

2.2 ViBE

The foreground object segmentation algorithm ViBE (Visual Background Extractor) was proposed by O. Barnich and M. Van Droogenbroeck in 2009 [1]. The background model in ViBE consists of a set of $N = 20$ observed pixel values. The classification is based on calculating the distance between the current pixel and all samples from the model. If at least $\#_{min} = 2$ distances are smaller than a given threshold the pixel is regarded as background. The model is updated by replacing randomly selected samples with the current pixel. Also a neighbourhood update procedure is proposed – a randomly selected sample from a random model in the 3×3 context is updated. To the best knowledge of the authors, a GPU implementation of ViBE has not been described. An FPGA implementation of this algorithm was presented in [7].

2.3 PBAS

The PBAS method [5] is an extension of the approach used in ViBE. Two additional parameters R and T were added to the algorithm. The first defines the threshold for the distance between the current pixel and samples from the model. In contrast to the ViBE, it is determined for each pixel independently. The second parameter specifies the probability of the model update. Also, information about edges was added to the background model. All modifications allowed to obtain slightly better segmentation results than ViBE. On the other hand, the background model of PBAS is larger than ViBE, as additional parameters need to be stored. To the best knowledge of the authors, the GPU implementation of PBAS has not been described. An FPGA implementation of this algorithm was presented in [6].

3 The Used Hardware Platforms

The methods described in Section 2 have been implemented on three platforms: CPU, GPU and FPGA.

General purpose processors are, and will remain for a long time, the most popular platform of implementation for all kinds of algorithms, including video processing. This is due to their vast availability and dissemination of the required programming skills. They are also very versatile, allowing to implement all kinds of algorithms from simple filtering, to machine learning methods. Unfortunately, in image processing tasks CPUs prove to be quite slow, as they are more suitable to perform complex instructions on small or medium datasets, but not simple instructions on huge amounts of data.

Graphic processing units have become a very popular platform for the realisation of parallel computing in recent years, mainly due to changes in architecture (unified shader) and development of CUDA and OpenCL languages. They allow to implement lots of image processing operations: simple filtration and morphological processing, Hough transform, optical flow, analysis and segmentation of MRI images and foreground objects segmentation. It should be noted, however, that the GPU architecture imposes certain restrictions on the implemented algorithms. Firstly, there are no jump instructions. Another problem is the memory model, which makes it difficult to perform operations on shared resources (update necessary in ViBE and PBAS methods). The evaluated algorithms were described in the OpenCL language, which is a cross-platform standard for parallel programming. This allowed to run them on both: multi-core CPU and GPU.

FPGAs are a proven platform for parallel computing, including image processing and analysis. The huge number of logical resources (LUT elements, flip-flops), multipliers and block RAMs allows for the implementation of any type of parallelism (according to Flynn's classification). An important feature of FPGAs is the ability to work in embedded devices – e.g. smart cameras, where image processing and analysis takes place immediately after the acquisition. Furthermore, compared to CPU or GPU, they are characterized by a relatively low energy consumption.

4 The Obtained Results

4.1 Hardware Platforms

In the experiments the following platforms were used:

– a PC (laptop) with Intel®Core™ i7-3632QM (4-cores, 64-bit, nominal freq. 2.2 GHz, maximal 3.2 GHz) and NVIDIA GeForce GT 645M GPU (384 CUDA cores, freq. 710 MHz, PCI-E 3.0),
– ML 605 evaluation board with Virtex 6 FPGA device (XC6VLX240T) from Xilinx (ViBE implementation and in-circuit verification),
– VC 707 evaluation board with Virtex 7 FPGA device (XC7VX485T) from Xilinx (PBAS implementation and in-circuit verification).

4.2 The Impact of Hardware Architecture on Accuracy

The implementation of an algorithm on a platform different than CPU usually requires some modifications. A good example are fixed-point calculations on FPGA devices. For the GMM method realized in OpenCL no changes were required and in the FPGA implementation fixed-point representation was used [3].

However, when implementing ViBE and PBAS on GPU, the neighbourhood update procedure proved to be problematic. The use of shared memory calculations (atomic operations) resulted in poor performance. It was decided, therefore, to modify the algorithm so that the accessing of neighbouring pixels was not required, while maintaining the update functionality. The step, in which for every pixel considered as background a randomly chosen adjacent model was updated, was replaced by the following mechanism. If in the neighbourhood of the considered pixel, at least one pixel labelled as background existed, than a randomly selected sample from the model was replaced by an adjacent value (selected from those regarded as background). This process occurred with a random probability. In addition, it was decided to use a fixed array of pre-generated pseudo-random numbers instead of their runtime generation.

The CIE Lab colourspace was used in the FPGA implementation of the ViBE method [7]. The number of samples in the model was reduced from 35 (CPU, GPU) do 19 (FPGA) due to hardware limitations (bandwidth to external RAM, in which the background model was stored) in the case of PBAS. Also the edges where not used in the model [6]. For both methods, all calculations were performed on fixed-point numbers.

The *changedetection.net 2012* dataset [4] was used to evaluate the impact of the proposed modifications. The obtained foreground masks were compared with manually annotated groundtruths and the parameters TP, TN, FP, FN were calculated. Then a number of common factors: Re (Recall), Sp (Specifity), FPR (False Positive Rate), FNR (False Negative Rate), PWC (Percent Wrong Clasiffication), F (F-measure), Pr (Precision) was computed. A detailed description of the methodology can be found in [6]. As post-processing median filtering

Table 1. Segmentation quality evaluation for different implementations of GMM, ViBE and PBAS methods

	Re	Sp	FPR	FNR	PWC	F	Pr
GMM	0.669	0.987	0.013	0.023	3.286	0.625	0.697
$\text{GMM}_{\text{OpenCL}}$	0.670	0.987	0.013	0.023	3.279	0.626	0.698
PBAS	0.837	0.922	0.018	0.108	1.994	0.637	0.799
$\text{PBAS}_{\text{OpenCL}}$	0.816	0.957	0.018	0.118	2.918	0.637	0.793
$\text{PBAS}_{\text{FPGA}}$	0.797	0.972	0.028	0.202	3.435	0.681	0.699
ViBE	0.723	0.969	0.031	0.012	3.846	0.677	0.761
$\text{ViBE}_{\text{OpenCL}}$	0.723	0.923	0.037	0.007	3.669	0.572	0.721
$\text{ViBE}_{\text{FPGA}}$	0.640	0.985	0.015	0.360	3.066	0.680	0.821

with 5×5 window size was used in all cases. The results are summarized in Table 1. The numbers for the GMM_{FPGA} version are not presented, as the authors of paper [3] did not use this dataset for evaluation.

An analysis of the results reveals that modifications related to hardware platform do not significantly affect the segmentation results. In case of ViBE and PBAS, there is a slight deterioration of the parameters associated with the fixed-point representation and limited numbers of samples (PBAS) in the FPGA version.

4.3 Comments Regarding the Implementations

During the implementation of the considered algorithms on the GPU, it turned out that comparing background variants (samples) with the current pixel is quite challenging. Since there is no support for the jump instruction, it is necessary to perform all comparisons (eventually substituted by an empty instruction – *nop*). Therefore, it is impossible to reduce the complexity by dynamically assigning the number of variants in case of the GMM method. The calculations will always be performed for the maximum number of variants ($K = 4$ in the experiments). However, the value is rather small and therefore the adverse effect is minimal.

A similar problem occurs in the case of the ViBE and PBAS methods. The number of variants (samples) is, however, much higher (20 and 35) and the effect is much more apparent. It is worth noting that such problems do not occur in the case of the CPU and FPGA implementations. In the first case, the jump instruction is supported, and in the second the architecture allows to perform all comparisons in parallel.

In the OpenCL language the access to individual memory bytes is slower than to groups of 4, 8 or 16 blocks. One possible solution is to convert the typical *RGB* format (3 bytes) to *0RGB* (4 bytes). This results in acceleration, but in the case of processing images from a camera an additional operation is required.

4.4 The Possibility of Real-Time Image Processing

In case of most video systems, real-time processing is very important. Good examples are surveillance of public space or traffic applications. The system operates in real-time when the sequence is processed without any pixel or frame dropping for a given image resolution and number of frames per second (*fps*). Most of contemporary cameras registers 25(30) or 50(60) fps.[2] In this study it was assumed that 25 *fps* is the reference value. The *fps* values for the analysed algorithms and implementations are summarized in Table 2.

Table 2. The number of frames that can be processed by different implementations in one second. 720 HD – resolution 1280 × 720, 1080 HD – resolution 1920 × 1080.

	CPU	$\mathrm{CPU_{OpenCL}}$	$\mathrm{GPU_{OpenCL}}$	FPGA
GMM 720 HD	10	13	63	205
GMM 1080 HD	5	6	27	91
ViBE 720 HD	2	7	3	151
ViBE 1080 HD	1	3	2	67
PBAS 720 HD	2	4	3	125
PBAS 1080 HD	1	2	-	55

The values for all methods in the column FPGA are estimations and were determined using the maximum clock frequency for the designed modules. In case of a working system, a cooperation with external RAM is required (storing the background model). This is often the "bottleneck" of the vision system. FPGA evaluation boards ML605 and VC707 available to the authors allowed to verify the algorithms ViBE for a video stream 640 × 480 @ 60 *fps* and PBAS for 720 × 576 @ 50 *fps*. In the first case the resource usage was: flip-flops 12571 (3%)[3], LUT6 9278 (6%), DSP48 3 (1%), BRAM_18 172 20% and in the second: flip-flops 39108 (6%), LUT6 36060 (11%), DSP48 12 (1%), BRAM_18 3 (1%), BRAM_36 248 (24%). The GMM method was verified on a Virtex 4 based platform for 1280 × 720 @ 20 *fps* video stream by the authors of paper [3]. The resource usage for the algorithm only (no video processing system with external RAM): flip-flops 363 (0.3%), LUT 788 (2%), DSP48 3 (1%), BRAM_18 0 (0%). On the GPU test platform, the PBAS $\mathrm{GPU_{OpenCL}}$ version for 1080 HD resolution could not be run due to lack of available memory. It should be pointed out that for FPGA, the number of processed frames per second is independent of the image content and other conditions. For other platforms the *fps* value is an average, which depends on the image content as well as operation of the computer system (e.g. operating system).

Analysis of the results allows to point out a difference between GMM and ViBE/PBAS methods. The first is ideally suited for GPU acceleration (×6 speed-up). The GPU implementation of methods with neighbourhood operations and

[2] The number is brackets refer to US video standards.

[3] % of available resources.

large number of samples in the model did not result in any acceleration. It seems that the main issue is the lack of the jump instruction. A further proof is the comparison of the $\mathrm{CPU}_{\mathrm{OpenCL}}$ and $\mathrm{GPU}_{\mathrm{OpenCL}}$ versions. The OpenCL implementation evaluated on a CPU turned out to be more efficient for ViBE and PBAS.

FPGA devices provide a wide opportunity to implement parallel computations. However, the bottleneck in this case is the access to external RAM memory resources. Another drawback is also the quite long development time in hardware description languages (i.e. Verilog).

4.5 The Concept of Using GPU and FPGA in a Vision System

The characteristic features of GPU and FPGA devices determine the way how they can be used in a vision system. In the first case, a close cooperation with the CPU within the computer system is necessary. Thus, the GPU could be an accelerator for certain computing tasks such as foreground segmentation.

The advantages of this solution are: the availability of the equipment and its low price, relative ease of programming and high parallel computing performance. In contrast, the main disadvantage is the need to implement data exchange between CPU and GPU. This is especially important in video processing, where the data stream is quite large. The measured transfer time ranges from 4% to 18% of the total computing time for GMM method and is less than 4% for ViBE and PBAS. In addition, the computer system must be equipped with an element responsible for the image acquisition. This could be, for example, a HDMI frame-grabber or network card for IP cameras.

The FPGA cooperating with an external RAM memory module can work independently of a processor system (standalone). Thus, in addition to the use as an accelerator (e.g. FPGA board with PCI-E communication), it can be used in a smart camera and in a configurable frame-grabber (HDMI or Ethernet). Also, the often required video stream decompression (H.264, MJPEG) could also be implemented in FPGA. This approach allows to reduce the required data transfers between different components of the system.

5 Conclusions

The article presents a comparison of three advanced foreground objects segmentation methods: GMM, ViBE and PBAS implemented on CPU, GPU and FPGA. It is shown that the implementation on GPU or FPGA has a very little impact on the segmentation quality, even if some modifications to algorithm were required. The analysis of the calculated *fps* values indicates that the GMM method is better suited for implementation in GPU than ViBE or PBAS. This is due to the relatively small number of distributions (variants) in the model ($K = 4$) and the independent processing of each pixel. The implementation of all methods is possible in FPGA and in all cases allows to obtain a significant acceleration. Here, the main limitation is the fast and wide access to external

RAM resources. FPGAs are more versatile because they can be used as an accelerator, part of smart-camera or an advanced frame-grabber. Also they are more energy efficient. However, they are not so widely available as GPUs and the process of implementation in hardware description language is usually quite time-consuming. In conclusion, before choosing a particular hardware platform to accelerate a segmentation algorithm the following issues should be considered: the computations present in the algorithm (especially involving neighbourhood operations) and the number of variants in the model as well as the architecture of the vision system.

Acknowledgments. The work presented in this paper was supported by AGH Univeristy of Science and Technology project number 11.11.120.612.

References

1. Barnich, O., Van Droogenbroeck, M.: ViBE: A Universal Background Subtraction Algorithm for Video Sequences. IEEE Transactions on Image Processing 20(6), 1709–1724 (2011)
2. Bouwmans, T., Porikli, F., Horferlin, B., Vacavant, A.: Handbook on: Background Modeling and Foreground Detection for Video Surveillance: Traditional and Recent Approaches, Implementations, Benchmarking and Evaluation. Taylor and Francis Group (2014)
3. Genovese, M., Napoli, E.: ASIC and FPGA Implementation of the Gaussian Mixture Model Algorithm for Real-Time Segmentation of High Definition Video. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 22(3), 537–547 (2014)
4. Goyette, N., Jodoin, P., Porikli, F., Konrad, J., Ishwar, P.: Changedetection.net: A new change detection benchmark dataset. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1–8 (2012)
5. 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)
6. Kryjak, T., Komorkiewicz, M., Gorgon, M.: Hardware implementation of the PBAS foreground detection method in FPGA. In: Proceedings of the 20th International Conference Mixed Design of Integrated Circuits and Systems (MIXDES), pp. 591–596 (2013)
7. 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. 479–484 (2013)
8. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. xxiii+637+663 (1999)
9. Pham, V., Vo, P., Hung, V.T., Bac, L.H.: GPU Implementation of Extended Gaussian Mixture Model for Background Subtraction. In: IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), pp. 1–4 (2010)