Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2013, Article ID 129589, 8 pages
http://dx.doi.org/10.1155/2013/129589

Research Article

FPGA Implementation of Gaussian Mixture Model Algorithm for 47fps Segmentation of 1080p Video

Mariangela Genovese, Ettore Napoli, Davide De Caro, Nicola Petra, and Antonio G. M. Strollo

Department of Biomedical, Electronic and Telecommunications Engineering, University of Napoli Federico II, 80125 Napoli, Italy

Correspondence should be addressed to Mariangela Genovese; mariangela.genovese@unina.it

Received 31 October 2012; Accepted 7 January 2013

Academic Editor: Ashkan Ashrafi

Circuits and systems able to process high quality video in real time are fundamental in nowadays imaging systems. The circuit proposed in the paper, aimed at the robust identification of the background in video streams, implements the improved formulation of the Gaussian Mixture Model (GMM) algorithm that is included in the OpenCV library. An innovative, hardware oriented, formulation of the GMM equations, the use of truncated binary multipliers, and ROM compression techniques allow reduced hardware complexity and increased processing capability. The proposed circuit has been designed having commercial FPGA devices as target and provides speed and logic resources occupation that overcome previously proposed implementations. The circuit, when implemented on Virtex6 or StratixIV, processes more than 45 frame per second in 1080p format and uses few percent of FPGA logic resources.

1. Introduction

Real time background identification in high definition video streams is an important task in smart video systems [1–4]. Various algorithms that address the problem have been proposed in the scientific literature throughout the years [5–16].

Foreground detection algorithms based on temporal difference techniques [5–7] allow the detection of relevant objects (Foreground, Fg) by comparing two or more consecutive frames but are unreliable in presence of illumination changes.

More robust algorithms, named background subtraction algorithms [8–15] are based on the creation of a model that includes the static parts of the scene (Background, Bg). The Fg is detected by analyzing the difference between a frame and the model of the Bg. The Bg subtraction algorithms regularly update the Bg model and are resilient to slow changes of the scene. They differ for the considered model of the Bg and for the algorithm that takes care of model updating.

References [13–15] propose Bg subtraction algorithms based on statistical models.

In [14] the Bg is represented with a statistical model based on a single Gaussian distribution per pixel. This allows a pixel by pixel model of the Bg with larger variance for pixels that experience wide changes of lighting.

The Gaussian Mixture Model (GMM) algorithm has been proposed in [15]. It uses a statistical model in which the probability distribution of the luminosity of each pixel is modeled with a mixture of Gaussian distributions. The GMM algorithm provides good performances in both presence of illumination changes and multimodal Bg. A multimodal Bg is characterized by objects showing repetitive motions, for example, waves, moving leaves, or flickering light. When a pixel lies in a region where a repetitive motion occurs, its brightness oscillates between two or more values. This results in false Fg detection in most algorithms. On the contrary, when the GMM algorithm is used, the intensity distribution of the pixel is modeled using two or more Gaussian distributions and the problem of false Fg detection is mitigated.

The OpenCV (Open source Computer Vision software library developed by Intel [16]) provides a common base of computer vision instruments that is becoming the de facto standard for video/image processing. The Bg detection

algorithm included in the OpenCV is an optimized version of the GMM algorithm of [15]. It changes the initial learning phase for the Bg model, that is improved with respect to [15], and calculates the learning rate in a simplified yet efficient way. As a consequence, the OpenCV version of the GMM algorithm is both widely used and provides good performances.

1.1. Previous Hardware Implementations. In order to generate the updated Bg model, the GMM algorithm processes the video stream by computing a great number of parameters for each pixel of each frame, with a computational burden unfeasible for computers in real time applications. As an example, in [15], only a frame rates of 11–13 frame per second (fps) is obtained for frame size 160 × 120 on an SGI O2 workstation.

Real time video applications with larger frame size require a dedicated hardware architecture. Hardware processors have been proposed in [17–22]. Reference [17] proposes a circuit able to process video sequences with frame size 1024 × 1024 at 38 fps when implemented on a VirtexII FPGA platform. Processing capabilities of [17] are 39.8 Mega pixel per second (Mps). In [18] the design of an automated digital surveillance system running in real time on an embedded platform is presented. The segmentation unit of the circuit proposed in [18] is able to run at 83 MHz on VirtexII xc2pro30. In [19], the research group of [18] improves the memory throughput with respect to [18] by employing a memory reduction scheme. The improvement in memory throughput is paid with a reduction of the processing capability that is limited to 7.68 Mps.

Reference [20] proposes an OpenCV GMM algorithm implementation able to process 22 HD (1920 × 1080 pixels) frames per second when implemented on Virtex5 xc5vlx50 FPGA. The circuit of [20] is used in [21] in conjunction with a denoising block that implements erosion, dilation, opening, and closing.

A circuit that implements the OpenCV GMM algorithm and provides increased performances has been presented in [22]. The improvement in circuit performance derives from the optimization of the bit width of the signals and from a better formulation of the GMM equations.

1.2. Proposed Circuit. In this paper a hardware implementation of the GMM algorithm that allows real time processing of HD videos and comply with the OpenCV algorithm is proposed. To the best of author's knowledge, the proposed circuit is the one with highest performance proposed to date. The circuit processes gray scale videos. When color videos need to be processed, the circuit, as reported in [19], can be fed with the luminance channel of the YCrCb color space.

The proposed circuit is optimized for an FPGA implementation and, with respect to the best performing previous work, [22], increases the maximum working frequency (+77.4%) and reduces the area utilization (−3.8% of Slices and 0 versus 10 DSP).

The improved performances are obtained by

(i) replacing full-width binary multipliers with truncated binary multipliers [23–26];

(ii) approximating nonlinear functions with piecewise linear approximations [27–31];

(iii) exploiting an innovative formulation of the updating equations of the GMM algorithm.

It is worth noting that the implemented design techniques (mainly the use of truncated multipliers and the approximation of nonlinear functions with piecewise linear functions) while allowing improved performances in terms of circuit speed and logic resources occupation are a modification of the reference OpenCV GMM algorithm and introduce a computation error that could result in a worsening of the quality of the processed video. The tests presented in Section 5, however, demonstrate that the proposed circuit is able to obtain very high performances while providing a good quality of the processed video sequences.

2. GMM Algorithm

The GMM algorithm has been proposed by Stauffer and Grimson, [15], with the target of efficiently dealing with multimodal Bg by using a statistical model composed by a mixture of Gaussian distributions. The GMM algorithm has been modified and included in the OpenCV libraries. A short description of the OpenCV GMM algorithm is given in the following. The differences with respect to the algorithm of [15] are indicated in the text.

2.1. Statistical Model. The statistical model for each pixel of the video sequence is composed by a mixture of K Gaussian distributions, each one represented by four parameters: weight (w), mean (μ), variance (σ^2), and *matchsum*, a counter introduced in the OpenCV algorithm.

Gaussian parameters differ for each Gaussian of each pixel and change for every frame of the video sequence. They are therefore defined by three indexes (p, k, t), where "p" is the index for the pixel, "k" is the index for the Gaussian distribution, and "t" is for the frame. In the following the pixel index is omitted since the same operations are repeated for every pixel.

2.2. Parameters Update. When a frame is acquired, for each pixel, the K Gaussian distributions are sorted in decreasing order of a parameter named Fitness ($F_{k,t}$):

$$F_{k,t} = \frac{w_{k,t}}{\sigma_{k,t}}. \tag{1}$$

A match condition is checked with the K Gaussian distributions that model the pixel. The match condition is

$$M_k = 1 \quad \text{if } \left|(\text{pixel} - \mu_{k,t})\right| < 2.5\sigma_{k,t}. \tag{2}$$

Equation (2) establishes if the pixel can be considered part of the Bg. A pixel can verify (2) for more than one Gaussian. The Gaussian that matches with the pixel ($M_k = 1$) and has

the highest $F_{k,t}$ value is considered as the "matched distribution" and its parameters are updated as follows:

$$\begin{aligned}\mu_{k,t+1} &= \mu_{k,t} + \alpha_{k,t} \cdot (\text{pixel} - \mu_{k,t}),\\ \sigma^2_{k,t+1} &= \sigma^2_{k,t} + \alpha_{k,t} \cdot \left[(\text{pixel} - \mu_{k,t})^2 - \sigma^2_{k,t}\right],\\ w_{k,t+1} &= w_{k,t} - \alpha_w \cdot w_{k,t} + \alpha_w,\\ \text{matchsum}_{k,t+1} &= \text{matchsum}_{k,t} + 1.\end{aligned} \tag{3}$$

The parameter α_w is the learning rate for the weight while $\alpha_{k,t}$ is the learning rate for mean and variance. It is derived from α_w as follows:

$$\alpha_{k,t} = \frac{\alpha_w}{w_{k,t}}. \tag{4}$$

Equation (4) is characteristic of the OpenCV algorithm and differs from what is proposed in [15] where $\alpha_{k,t}$ is calculated, being η the Gaussian probability density function as follows:

$$\alpha_{k,t} = \alpha_w \cdot \eta(\text{pixel}, \mu_{k,t}, \sigma_{k,t}). \tag{5}$$

The approach of [15] results, as shown in [32], in a slow convergence if compared with (4).

For the unmatched Gaussian distributions mean and variance are unchanged while the weights are updated as follows:

$$w_{k,t+1} = w_{k,t} - \alpha_w \cdots w_{k,t}. \tag{6}$$

When the pixel does not match any Gaussians, a specific "no_match" procedure is executed and the Gaussian distribution with smallest $F_{k,t}$ is updated as follows:

$$\begin{aligned}\mu_{k,t+1} &= \text{pixel} \qquad \text{matchsum}_{k,t+1} = 1,\\ \sigma^2_{k,t+1} &= \text{vinit} \qquad w_{k,t+1} = \frac{1}{\text{msumtot}},\end{aligned} \tag{7}$$

where vinit is a fixed initialization value and msumtot is the sum of the values of the matchsum of the $K-1$ Gaussians with highest $F_{k,t}$. The weights of the $K-1$ Gaussians with highest $F_{k,t}$ are decremented as in (6) while their means and variances are unchanged.

2.3. Background Identification. The Bg identification uses the following algorithm:

$$B = \arg\min_b \left(\sum_{k=1}^{b} w_{k,t} > T \right). \tag{8}$$

Equation (8) adds in succession the weights of the first b Gaussian distributions, sorted beginning from the one with highest $F_{k,t}$ value, until their sum is greater than T, a fixed threshold belonging to the $[0, 1]$ interval. The set of Gaussian distributions that verify (8) represents the Bg and a pixel that matches one of these Gaussians is classified as a Bg pixel. The algorithm entails that if the "no_match" condition occurs, the pixel is classified as Fg.

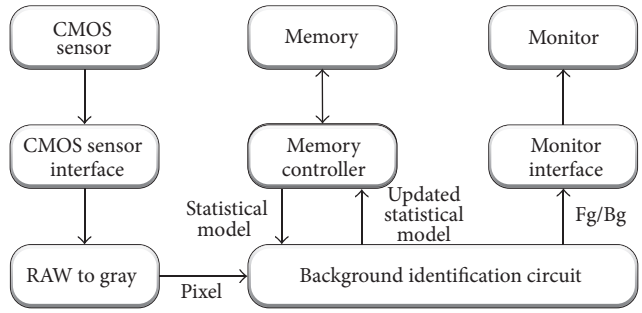

Figure 1: Conceptual overview of a background identification system.

3. Background Identification Circuit

Figure 1 shows a conceptual overview of a Bg identification system. A CMOS sensor captures each frame of the video sequence and a CMOS interface gives the pixel values to the Bg identification circuit. The Bg identification circuit processes the luminance value of the input "Pixel," and the "Statistical Model" of the pixel for the given frame. The output data are the "Updated Statistical Model" and the "Fg/Bg" tag. For each pixel, the Gaussian parameters are read from an external memory and the updated parameters are stored in the memory. For each frame of the input video sequence, the Fg/Bg tags produce a binary image that can be displayed on a monitor.

In this paper, the "background identification circuit" of Figure 1 has been implemented on FPGA devices. Figure 2 shows the block diagram of the proposed circuit. The circuit is described in VHDL code and implements the GMM algorithm using, as suggested in [15], three Gaussian distributions for each pixel ($k = [1, 2, 3]$). As a consequence, the proposed circuit processes 13 parameters per pixel (the luminance value and 12 Gaussians parameters for the statistical model of the pixel).

The software algorithm of the OpenCV library represents the Gaussians parameters as double precision (64 bit) floating point numbers. A double precision hardware implementation would be impractically large and slow and would require a very high bandwidth towards the memory. The proposed circuit represents the signals as fixed point numbers on a limited number of bits. The word length and the representation of the Gaussians parameters are determined by using the word length optimization algorithm proposed in [22] that minimizes the word length while constraining the Peak Signal-to-Noise Ratio (PSNR) of a set of test video streams. The input bits of the resulting circuit are the 99 bits for the parameters of the statistical model and the 8 bits of the luminance of the input pixel. The output bits are the updated statistical model, that requires 99 bits, and the Fg/Bg tag.

The optimized fixed point representation, an innovative formulation of the GMM algorithm equations (described in Section 4), the use of truncated multipliers, and the application of ROM optimization techniques (details in Section 5),

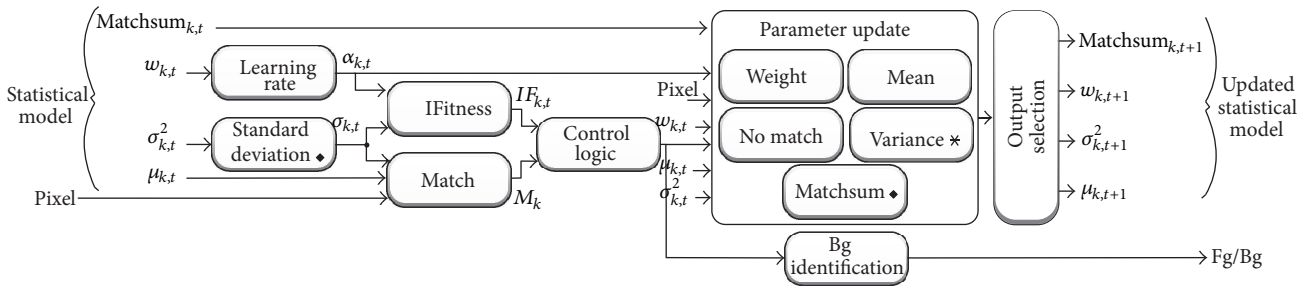

Figure 2: Block diagram of the proposed FPGA implementation. The functionality of the circuit is described in Section 5. The star indicates the circuital block that has been improved by using the truncated binary multipliers. The diamond indicates the circuital blocks that have been improved by using piecewise linear approximation of nonlinear functions.

allow the design of a Bg identification circuit with high working frequency and low logic resources utilization.

4. Hardware Oriented Formulation of the GMM Equations

The equations of the GMM algorithm, detailed in Section 2, if straightforwardly implemented, would be too complex for the realization of a fast and compact circuit. The proposed circuit is therefore based on the manipulation and simplification of the above-cited equations. A first step towards the simplification is the use of an optimized fixed point representation for circuit signals [22]. A second step is the smart implementation of the nonlinear operations as will be shown in Section 5.

In addition to the above-cited techniques, in this paper, in order to reduce the logic occupation, a new order parameter for Gaussian distributions is introduced. The parameter is named *IFitness* (indicated in the following as $IF_{k,t}$) and is defined as the inverse of the $F_{k,t}$ factor:

$$IF_{k,t} = \frac{1}{F_{k,t}}. \tag{9}$$

With simple manipulations, taking into account (1) and (4), $IF_{k,t}$ can be written as

$$IF_{k,t} = \frac{\sigma_{k,t}}{w_{k,t}} = \frac{\sigma_{k,t}}{w_{k,t}} \cdot \frac{\alpha_{k,t}}{\alpha_{k,t}} = \sigma_{k,t} \cdot \frac{\alpha_{k,t}}{\alpha_w}. \tag{10}$$

Equation (10) shows that the $IF_{k,t}$ factor can be computed as a function of the learning rate $\alpha_{k,t}$.

Moreover, in the proposed implementation, the learning rates α_w and $\alpha_{k,t}$ are quantized as power of two values:

$$\alpha_w = 2^{n_w} \qquad \alpha_{k,t} = 2^{n_{k,t}}. \tag{11}$$

As a consequence, the $IF_{k,t}$ factor can be computed as follows:

$$IF_{k,t} = \sigma_{k,t} \cdot 2^{(n_{k,t} - n_w)} \tag{12}$$

and the multiplication between $2^{(n_{k,t} - n_w)}$ and $\sigma_{k,t}$ can be performed using a shifter instead of a binary multiplier.

It is worth noting that ordering the Gaussian distributions as a function of $F_{k,t}$, a process needed in the GMM algorithm in order to determine the Bg model according to (8), is equivalent to ordering versus the $IF_{k,t}$ value.

Equations (4) and (12) can be implemented using only a shifter and a square root circuit that calculates the standard deviation $\sigma_{k,t}$. The implementation does not require the multiplier that would be needed to implement (1). The removal of the multiplier is not only beneficial for the implied reduction of hardware usage but, being the multiplier on the critical path of the Bg identification circuit, is also effective in increasing overall circuit speed. More details regarding the HW implementation of the above-described equations are given in the Section 5.

5. FPGA Implementation of the Background Identification Circuit

The schematic of the FPGA implementation of the GMM algorithm is shown in Figure 2. A detailed explanation of the functionalities and of the hardware implementation of the circuital blocks of Figure 2 is given in the following.

5.1. Learning Rate. Computes $\alpha_{k,t}$ following (4).

The "learning rate" unit is implemented with three ROM, each of them stores the power of two values that better approximates $\alpha_{k,t}$ as a function of $w_{k,t}$ ((4), (11)). The equation implemented by the ROM is

$$n_{k,t} = \text{round}\left[\log_2\left(\alpha_{k,t}\right)\right] \tag{13}$$

in which the function "round(·)" is the approximation to the nearest integer value. The $n_{k,t}$ values are then used by the "*IFitness*" unit to shift the $\sigma_{k,t}$ in order to calculate the $IF_{k,t}$ factor according to (12).

5.2. Standard Deviation. The formulation of the GMM equations described in Section 4 requires the standard deviation values, $\sigma_{k,t}$, for the three Gaussians while the input data for the circuit include the variance values, $\sigma^2_{k,t}$.

The "Standard Deviation" block determines the variance values by implementing a square root circuit.

In the proposed implementation the $\sigma^2_{k,t}$ signals are represented on 11 bits and range in [8 : 16376]. Due to the reduced number of bits of the variance signal, a ROM implementation for the square root circuit could be feasible but not optimal. In this paper the square root function has instead been implemented by using a piecewise linear approximation of the nonlinear function, a technique that has been previously proposed as an effective method to reduce hardware complexity [27–31].

The interval [8 : 16376] is divided, as shown in Figure 3, into four nonuniform intervals, $I_1, \ldots, I_4$. In each interval, a linear function y_i approximates $\sigma_{k,t}$ as follows:

$$y_i = q_i - m_i \cdot \sigma^2_{k,t} \quad \sigma^2_{k,t} \in I_i \tag{14}$$

with $i = 1, \ldots, 4$ and $q_i, m_i > 0$. In order to simplify the hardware, the m_i coefficients are quantized as powers of two and the multiplication between m_i and $\sigma^2_{k,t}$ is performed by using a shifter resulting in lower area utilization and higher working frequency.

5.3. IFitness. It computes $IF_{k,t}$ according to (12).

The circuit is actually a combinatorial shifter that shifts $\sigma^2_{k,t}$ as a function of the $n_{k,t}$ value and obtains the $IF_{k,t}$ factor.

The proposed formulation of the GMM equations, that use (4) and (12) computed by the "learning rate" and "*IFitness*" block, implemented on Virtex5 vlx50 FPGA requires 64 Slices. This result can be compared with the implementation of (4) and (1) (the technique used in [22]) that, on the same FPGA, requires 123 Slices and 3 DSP. This demonstrates that the new formulation of the GMM algorithm equations is effective in reducing circuit complexity.

5.4. Match. It verifies the match condition for the three Gaussian distributions and is composed by three identical units (one for each Gaussian). Each unit implements (15) and employs a subtractor and one comparator as follows:

$$M_k = 1 \quad \text{if } \left|\text{pixel} - \mu_{k,t}\right| < 2.5 \cdot \sigma_{k,t}. \tag{15}$$

5.5. Control Logic. It sorts the Gaussians in increasing $IF_{k,t}$ order and establishes which Gaussian has to be updated as in (3), (6), and (7).

5.6. Parameter Update. This circuital block is composed by five subunits ("Weight", "Mean", "Variance" "No Match", and "Matchsum") that implement the parameters update equations (3), (6), and (7) detailed in Section 2.

5.6.1. "Weight" and "Mean". If the match condition is verified, these blocks update the parameters according to (3) and (6). The quantization of the learning rates as power of two allows to implement (3) by using shifters instead of multipliers.

5.6.2. "Variance". The left hand of the variance update equation of (3) requires the computation of a square. In the proposed circuit the square has been implemented by using a truncated multiplier [26].

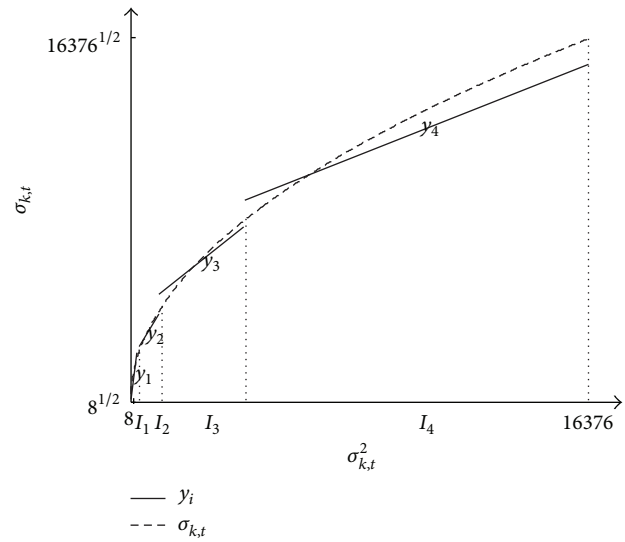

Figure 3: Piecewise linear polynomial approximation of the $\sigma_{k,t}$ function implemented in the "Standard Deviation" block of Figure 2.

Truncated binary multipliers, for which the optimality in terms of mean square error is analytically demonstrated, have been proposed in [23–26]. Truncated multipliers have been used with good results in a variety of applications [33–35]. A full-width digital $n \times n$ bits multiplier computes the $2n$ bits output as a weighted sum of partial products. A truncated multiplier is an $n \times n$ multiplier with an $m < 2n$ bits output.

In the proposed circuit the inputs of the multipliers are represented on 12 bits. The output of a 12×12 full-width multiplier is hence on 24 bits. In order to reduce HW complexity and improve circuit speed, the proposed circuit implements a truncated binary multiplier, based on the architecture proposed in [24], in which the output is directly produced on 15 bits (9 less-significant bits are discarded with an almost 40% reduction of the required HW for the multiplier).

5.6.3. "No Match". When no Gaussians match the pixel, the "No Match" block updates the parameters of the Gaussian with highest *IFitness* value according to (7). Equation (7) requires the calculation of the inverse of the msumtot signal. In [22] a ROM-based approach is used to implement the required function. Since the msumtot signal is represented on 6 bits and its inverse is on 8 bits, the ROM size is equal to $2^6 \times 8$.

In the proposed paper the ROM is eliminated by using a piecewise linear approximation technique similar to those used for the calculation of the $\sigma_{k,t}$ signals. The msumtot signal range [1 : 63] is divided into four nonuniform intervals J_i and in each interval, a function z_i approximates 1/msumtot as follows:

$$z_i = \lambda_i - \eta_i \cdot \text{msumtot}, \quad \text{msumtot} \in J_i, \tag{16}$$

with $i = 1, \ldots, 4$, λ_i, $\eta_i > 0$, and η_i quantized as power of two so that the multiplication between η_i and msumtot is performed using a shifter.

Table 1: Performances of the circuit of Figure 2 implemented on different FPGA and comparison with previous art.

Target FPGA	Circuit	LUT/ALUT	Flip Flop	Slice/LAB	DSP/MULT	Frequency (MHz)	HD fps
Virtex6 (xc6vlx75t)	proposed circuit	922/46560	0/93120	265/11640	0/288	97.19	46
Virtex5 (xc5vlx50)	proposed circuit	844/28800	0/28800	301/7200	0/48	91.03	43
	Reference [22]	705/28800	0/28800	313/7200	10/48	51.30	24
StratixIV (SGX230HF35C2)	proposed circuit	1150/182400	0/182400	81/9120	0/1288	98.12	47
Spartan3 (xc3s1000)	proposed circuit	1797/15360	0/15360	1010/7680	0/24	35.30	17

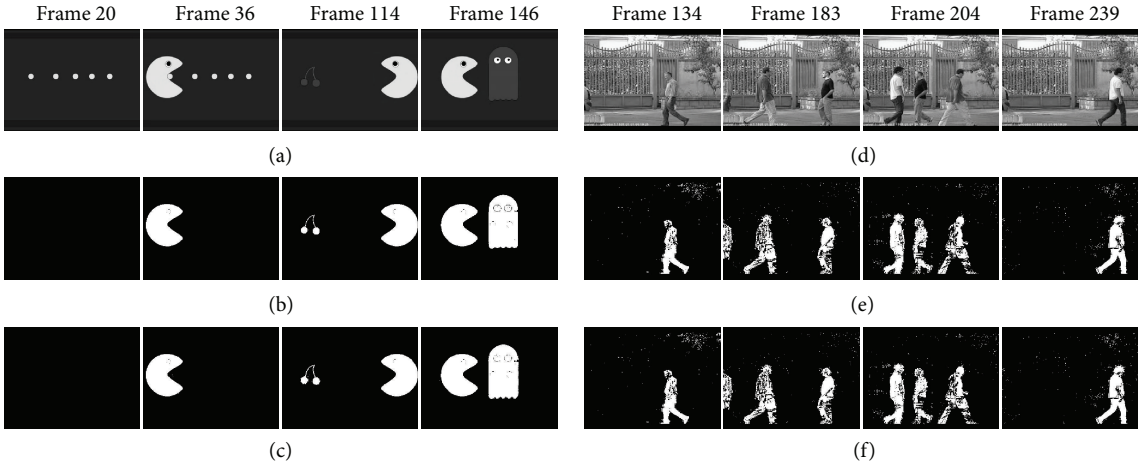

Figure 4: Comparison between the Bg masks obtained with the circuit of [22] and with the proposed implementation. (a), (d) unprocessed video frames; (b), (e) output of the proposed circuit; (c), (f) output of the circuit proposed in [22].

5.6.4. "Matchsum". This is the circuit that updates the *matchsum* signal according to (3) and (7). The matchsum signal is a counter, associated to every Gaussian, introduced in the OpenCV with the aim to improve the initial learning phase of the GMM algorithm (as demonstrated in [22]). The matchsum signal counts the number of times that a given Gaussian is matched according to (2). The introduction of the matchsum entails the synthesis of three counters that are not on the critical path. The only drawback is an increase of circuit area.

5.7. Output Selection. It establishes the values of the updated parameters depending on whether the match condition is verified or not.

5.8. Bg Identification. It verifies the background identification condition shown in (8) determining, also as a function of the "No Match" condition, if the input pixel belongs to the background or not.

6. Results

6.1. CAD Tools for Design and Test. The proposed FPGA implementation of the OpenCV GMM algorithm is synthesized and implemented on Virtex6 (xc6vlx75t), Virtex5 (xc5vlx50), and Spartan3 (xc3s1000) Xilinx FPGA and on Stratix IV (SGX230HF35C2) Altera FPGA. The performances of the proposed circuit on FPGA devices and the comparison with previous art are shown in Table 1.

For Virtex6, Virtex5, and Spartan3 the synthesis has been carried out using XST and fitting and Place&Route have been carried out using ISE. The implementation on StratixIV has been conducted with Quartus II. ModelSim PE has been used to perform the circuit simulations.

6.2. Quality of the Processed Video Streams. The implementation of (12), the utilization of truncated multipliers, and the implementation of ROM approximation techniques could worsen the accuracy of the processed video streams. Figure 4 shows a selection of frames with resolution 320 × 240 extracted from two video sequences. The original frames of Figures 4(a) and 4(d) have been processed by using the proposed implementation (Figures 4(b) and 4(e)) or by using the circuit of [22] that uses full-width multipliers and no ROM compression techniques (Figures 4(c) and 4(f)). The Bg masks of Figures 4(b), 4(e), and 4(c), 4(f) differ in few pixels (in both the video sequences less than the 1% of the total number of the pixels is differently classified by the two circuits). This demonstrates that the proposed implementation provides negligible effect on the quality of the processed video streams.

6.3. Circuit Performances. Table 1 shows that the circuit of Figure 2, implemented on high-performances FPGA devices, is able to process more than 43 HD fps by using less than the 5% of the FPGA resources. Implemented on Virtex6 processes 46 HD fps by using the 2% of the total number of Slice of the FPGA. On Virtex5 the frame rate is 43 HD fps

while the logic utilization is equal to 301 of 7200 Slices, when implemented on StratixIV processes 47 HD fps by using less than the 1% of Logic Adaptive Blocks (LAB).

The performances analysis has been conducted by including input and output registers that synchronize the circuits and provide timing performances that are not dependent on the I/O pads. The reported frame rate values are obtained by considering the maximum work frequency of the circuits (Frequency in Table 1), obtained by imposing a constraint on the minimum period of the clock signal. For each implementation of Table 1, the synthesis of the circuit has been carried out by repeatedly reducing the constraint. The process has been stopped when a 0.1 ns reduction made the implementation impossible. The last synthesized clock period has determined the maximum work frequency of the circuit. The timing analysis of the obtained implementations has allowed to identify the critical path of the circuit that includes "Match," "Control Logic," "Variance," and "Output Selection" units of Figure 2. However, the very closed constraint on the minimum clock period entails that all the paths between inputs and outputs of the circuit have delay values almost equal to the maximum delay.

The previous art of the background identification circuits is represented by [22] that provides improved performances if compared against [17–21].

The proposed FPGA implementation, as shown in Table 1 outperforms the circuit proposed in [22]. Since that the proposed circuit improves many blocks that in [22] was on the critical path, when the proposed circuit is implemented on Virtex5 xc5vlx50, the maximum working frequency almost doubles (+77.4%), while the area utilization is strongly reduced (−3.8% in number of used Slices and zero DSP used versus 10 DSP).

Table 1 also shows the performances of the circuit on low-cost Spartan3 FPGA. Implemented on Spartan3, the proposed circuit is able to run at 35.30 MHz by using the 13% of the total number of Slice of the FPGA.

7. Conclusion

An OpenCV compatible hardware implementation of the GMM algorithm for Bg identification is presented in the paper. The circuit has been designed having commercial FPGA devices as a target and is able to process 1080p video at 47 frames per second when implemented on Statix IV FPGA.

The circuit exploits an innovative formulation of the equations of the GMM algorithm, approximates some nonlinear functions with piecewise linear polynomials, and uses truncated binary multipliers. The circuit is implemented on both Altera and Xilinx FPGA and, when compared with previously proposed Bg identification circuits, provides improved speed and reduced logic utilization.

References

[1] I. Haritaoglu, D. Harwood, and L. S. Davis, "A fast background scene modeling and maintenance for outdoor surveillance," in *Proceedings of the 15th International Conference on Pattern Recognition*, no. 4, pp. 179–183, Barcelona, Spain, 2000.

[2] P. KaewTrakulPong and R. Bowden, "A real time adaptive visual surveillance system for tracking low-resolution colour targets in dynamically changing scenes," *Image and Vision Computing*, vol. 21, no. 10, pp. 913–929, 2003.

[3] B. Gloyer, H. K. Aghajan, K. Y. S. Siu, and T. Kailath, "Video-based freeway-monitoring system using recursive vehicle tracking," in *Image and Video Processing III*, vol. 2421 of *Proceedings of SPIE*, pp. 173–180, San Jose, Calif, USA, February 1995.

[4] H. Jiemnez and J. Salas, "Temporal templates for detecting the trajectories of moving vehicles," in *Advanced Concepts for Intelligent Vision Systems*, vol. 5807 of *Lecture Notes in Computer Science*, pp. 485–493, 2009.

[5] C. Zhan, X. Duan, S. Xu, Z. Song, and M. Luo, "An improved moving object detection algorithm based on frame difference and edge detection," in *4th International Conference on Image and Graphics, ICIG 2007*, pp. 519–523, Chengdu, China, August 2007.

[6] M. Ren and H. Sun, "A practical method for moving target detection under complex background," *Computer Engineering*, vol. 31, no. 20, pp. 33–40, 2005.

[7] B. Xiao, C. Lu, H. Chen, Y. Yu, and R. Chen, "Moving object detection and recognition based on the frame difference algorithm and moment invariant features," in *Proceedings of the 27th Chinese Control Conference (CCC '08)*, pp. 578–581, July 2008.

[8] S. Y. Chien, S. Y. Ma, and L. G. Chen, "Efficient moving object segmentation algorithm using background registration technique," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 7, pp. 577–586, 2002.

[9] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and foreground detection using Kalman filtering," in *Proceedings of the 10th International Congress for Applied Mineralogy (ICAM '95)*, pp. 193–199, 1995.

[10] K. P. Karmann and A. Brandt, "Moving object recognition using an adaptive background memory," in *Time-Varying Image Processing and Moving Object Recognition*, V. Capellini, Ed., pp. 289–307, Elsevier, 2nd edition, 1990.

[11] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in *Proceedings of the 1999 7th IEEE International Conference on Computer Vision (ICCV '99)*, pp. 255–261, September 1999.

[12] J. C. S. Jacques, C. R. Jung, and S. R. Musse, "Background subtraction and shadow detection in grayscale video sequences," in *Proceedings of the 18th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI '05)*, pp. 189–196, October 2005.

[13] C. C. Chiu, M. Y. Ku, and L. W. Liang, "A robust object segmentation system using a probability-based background extraction algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 4, pp. 518–528, 2010.

[14] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[15] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '99)*, pp. 246–252, Fort Collins, Colo, USA, June 1999.

[16] OpenCV library on source forge, http://sourceforge.net/projects/opencvlibrary/.

[17] H. Jiang, H. Ardo, and V. Öwall, "Hardware accelerator design for video segmentation with multi-modal background modelling," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 2, pp. 1142–1145, 2005.

[18] F. Kristensen, H. Hedberg, H. Jiang, P. Nilsson, and V. Öwall, "An embedded real-time surveillance system: implementation and evaluation," *Journal of Signal Processing Systems*, vol. 52, no. 1, pp. 75–94, 2008.

[19] H. Jiang, H. Ardö, and V. Öwall, "A hardware architecture for real-time video segmentation utilizing memory reduction techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 2, pp. 226–236, 2009.

[20] M. Genovese, E. Napoli, and N. Petra, "OpenCV compatible real time processor for background foreground identification," in *Proceedings of the International Conference on Microelectronics (ICM '10)*, pp. 467–470, December 2010.

[21] M. Genovese and E. Napoli, "FPGA-based architecture for real time segmentation and denoising of HD video," *Journal of Real-Time Image Processing*, 2011.

[22] M. Genovese and E. Napoli, "FPGA implementation of OpenCV compatible background identification circuit," in *Proceedings of the 3rd International Symposium on Computational Modelling of Objects Represented in Images: Fundamentals, Methods and Applications (CompIMAGE '12)*, pp. 75–80, Rome, Italy, September 2012.

[23] A. G. M. Strollo, N. Petra, and D. De Caro, "Dual-tree error compensation for high performance fixed-width multipliers," *IEEE Transactions on Circuits and Systems II*, vol. 52, no. 8, pp. 501–507, 2005.

[24] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 6, pp. 1312–1325, 2010.

[25] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Designof fixed-width multipliers with linear compensation function," *IEEE Transactions on Circuits and Systems I*, vol. 58, no. 5, pp. 947–960, 2011.

[26] V. Garofalo, N. Petra, and E. Napoli, "Analytical calculation of the maximum error for a family of truncated multipliers providing minimum mean square error," *IEEE Transactions on Computers*, vol. 60, no. 9, pp. 1366–1371, 2011.

[27] D. De Caro, E. Napoli, and A. G. M. Strollo, "Direct digital frequency synthesizers using high-order polynomial approximation," in *Proceedings of the IEEE International Solid-State Circuits Conference*, vol. 1, pp. 134–135, San Francisco, Calif, USA, February 2002.

[28] D. De Caro, E. Napoli, and A. G. M. Strollo, "Direct digital frequency synthesizers with polynomial hyperfolding technique," *IEEE Transactions on Circuits and Systems II*, vol. 51, no. 7, pp. 337–344, 2004.

[29] D. De Caro and A. G. M. Strollo, "High-performance direct digital frequency synthesizers in 0.25 μm CMOS using dual-slope approximation," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 11, pp. 2220–2227, 2005.

[30] A. G. M. Strollo, D. De Caro, and N. Petra, "Elementary functions hardware implementation using constrained piecewise-polynomial approximations," *IEEE Transactions on Computers*, vol. 60, no. 3, pp. 418–432, 2011.

[31] D. De Caro, N. Petra, and A. G. M. Strollo, "High-performance special function unit for programmable 3-D graphics processors," *IEEE Transactions on Circuits and Systems I*, vol. 56, no. 9, pp. 1968–1978, 2009.

[32] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Proceedings of the 2nd European Workshop Advanced Video Based Surveillance Systems*, 2001.

[33] V. Garofalo, N. Petra, D. De Caro, A. G. M. Strollo, and E. Napoli, "Low error truncated multipliers for DSP applications," in *Proceedings of the 15th IEEE International Conference on Electronics, Circuits and Systems (ICECS '08)*, pp. 29–32, September 2008.

[34] S. S. Kidambi and P. El-Guibaly, "Area-efficient multipliers for digital signal processing applications," *IEEE Transactions on Circuits and Systems II*, vol. 43, no. 2, pp. 90–95, 1996.

[35] V. Garofalo, "Fixed-width multipliers for the implementation of efficient digital FIR filters," *Microelectronics Journal*, vol. 39, no. 12, pp. 1491–1498, 2008.