J Real-Time Image Proc (2013) 8:389–401
DOI 10.1007/s11554-011-0238-1

SPECIAL ISSUE

FPGA-based architecture for real time segmentation and denoising of HD video

M. Genovese · E. Napoli

Received: 1 February 2011 / Accepted: 14 December 2011 / Published online: 28 December 2011

Abstract The identification of moving objects is a basic step in computer vision. The identification begins with the segmentation and is followed by a denoising phase. This paper proposes the FPGA hardware implementation of segmentation and denoising unit. The segmentation is conducted using the Gaussian mixture model (GMM), a probabilistic method for the segmentation of the background. The denoising is conducted implementing the morphological operators of erosion, dilation, opening and closing. The proposed circuit is optimized to perform real time processing of HD video sequences (1,920 × 1,080 @ 20 fps) when implemented on FPGA devices. The circuit uses an optimized fixed width representation of the data and implements high performance arithmetic circuits. The circuit is implemented on Xilinx and Altera FPGA. Implemented on xc5vlx50 Virtex5 FPGA, it can process 24 fps of an HD video using 1,179 Slice LUTs and 291 Slice Registers; the dynamic power dissipation is 0.46 mW/MHz. Implemented on EP2S15F484C3 StratixII, it provides a maximum working frequency of 44.03 MHz employing 5038 Logic Elements and 7,957 flip flop with a dynamic power dissipation of 4.03 mW/MHz.

Keywords Image motion analysis · Image segmentation · Morphological operations · High definition video · Field programmable gate arrays

M. Genovese (✉) · E. Napoli
DIBET, University of Napoli Federico II, Napoli, Italy
e-mail: mariangela.genovese@unina.it

E. Napoli
e-mail: etnapoli@unina.it

1 Introduction

Motion detection is one of the most important tasks in computer vision. Segmentation algorithms able to find moving objects (Foreground, *Fg*) in video sequences have been developed in the past and electronic systems have been implemented and employed in applications like video surveillance [1–4], and traffic monitoring [5–8].

A segmentation algorithm begins with an identification phase that detects objects supposedly belonging to the *Fg* [9]. The input data of a segmentation algorithm are a video sequence that can be both a grayscale image or a RGB image. The output of the segmentation algorithm is a video sequence composed of binary images in which a pixel is represented with one bit that is equal to '0' if the pixel is classified as background or '1' if the pixel is classified as foreground (or viceversa). The identification is usually followed by a denoising phase that uses morphological operators to remove noise and enhance the appearance of the binary images [10].

The literature reports various contributions for the initial segmentation of the *Fg* pixels [11–22]. Some of the segmentation algorithms, named background subtraction methods, compare the frame with a reference model (background, *Bg*) [14–21]. Target of these algorithms is the determination of the *Bg* model. Further, this model has to be updated to track the background scene.

Chien et al. [14] builds a *Bg* model starting with the assumption that, if a pixel is stationary for a prefixed number of consecutive frames, the probability that it belongs to the *Bg* is high and, therefore, the pixel can be included in the *Bg* model. Ridder et al. [15] and Karmann and Brandt [16] use a Kalmann filter while Toyama et al. [17] employs a Wiener filter. Jacques et al. [18] proposes an algorithm based on median filtering to adapt the *Bg* model.

Algorithms [15–18] take into account changes in the lighting of the *Bg* and the eventuality of moving objects that become static and exhibit low computational complexity.

Statistical *Bg* models have been proposed in Refs. [19–21]. The identification method, proposed in [19], models the fluctuations in a pixel value with a statistical model based on a single Gaussian distribution. This allows a pixel by pixel representation of the *Bg* with larger variance for pixels that experience wide change of lighting.

The algorithms in Refs. [15–18] and the statistical method of Wren et al. [19] are not suited to describe multimodal backgrounds in which a pixel oscillates between two or more values (e.g. moving leaves).

Stauffer and Grimson [20, 21] propose a statistical model, known as Gaussian mixture model (GMM), similar to [19] with the difference that each pixel is modeled with a mixture of Gaussian distributions to provide good performances in both presence of illumination changes and multimodal backgrounds.

The GMM is the *Bg* detection algorithm adopted in the OpenCV (Open Source Computer Vision) library [22] developed by Intel to provide a common base of computer vision instruments able to extract relevant details from the images and to process them in automatic way. The OpenCV library is becoming a widely used standard. At the time, the version 2.2 of the library is available, it includes more than 2,000 software algorithms, has been downloaded more than two million times and its user group is composed of more than 40,000 people. The importance of the OpenCV library is demonstrated by the fact that most (if not all of them) of the companies working on electronic systems for computer vision are now studying or implementing the OpenCV libraries to provide systems that can be easily transferred to new applications.

The GMM algorithm proposed in the OpenCV library modifies the algorithm of Stauffer and Grimson [20] to improve the initial learning phase.

Main drawback of the GMM algorithm is the number of complex non linear computations required for each pixel of the image to generate the updated *Bg* model. The calculations increase with the number of Gaussians per pixel and, fixed the number of Gaussian distributions, grows with the number of pixel per frame. The computational requirements, in conjunction with the target of processing high definition (HD) video streams, require a hardware implementation since a software implementation does not meet the required performances.

The actual trend is towards high performance systems with increased functionalities and towards lightweight and portable systems. The increased functionalities are, as example, higher video resolution and stereoscopic vision conjugated with the algorithmic smartness that makes the system able to adapt to variable conditions. The reduction in cost and the shift to portable systems are mainly targeted through the extensive use of integrated digital electronic systems and, very often, FPGA-based platforms.

High speed *Fg/Bg* processors have been proposed in Refs. [20, 23–36]. Between the referenced papers, it is possible to find implementations of the GMM algorithm that improve the method in various aspects such as scalability, quality of the segmented image, shadow removal, ghost detection, etc.

Stauffer and Grimson [20] proposes an implementation of the GMM algorithm able to process 11 fps for a small frame size of 160 × 120 on an SGI 02 workstation with a RIOOOO processor. The implementation proposed in [20] is. therefore, able to process 0.2 Mps (Mega pixels per second). Minghua and Bermak [23] proposes an ASIC implementation of a pattern recognition system based on a GMM classifier that, implemented in 0.25 μm technology is able to run at 80 MHz. Silicon area occupation is 1.69 mm^2. The GMM classifier of [23] is not applied to *Fg/Bg* identification but provides an indication of the expected performance for an ASIC implementation.

Varcheie et al. [24], Lin et al. [25] and Suhr et al.[26] propose improved GMM algorithms. The authors of [24] combine a GMM model with a region-based algorithm based upon color histograms and texture information. In their experiments, the proposed method outperforms the original GMM algorithm but with considerable computational cost. The algorithm proposed in [24] processes 2.2 Mps with an Intel Xeon 5150 processor.

Lin et al. [25] improves the GMM algorithm by adding further information to each processed pixel. This allows a different learning rate for each pixel and a better adaptation to the dynamic of the scenes. The paper shows that the quality of the segmented videos overcome the conventional GMM algorithm. No info is provided in [25] with respect to the computation requirement or the bandwidth of the method. It is expected that the required bandwidth is higher than what requested by the conventional GMM algorithm.

Suhr et al. [26] proposes a GMM algorithm that is able to directly process Bayer-pattern color images (avoids the RGB conversion). The proposed method requires a CPU time slightly larger than the conventional GMM and has been implemented on 2.8 GHz Intel I7 platform.

Kristensen et al. [27], Jiang et al. [28, 29], Minghua and Bermak [30] and Genovese et al. [31] propose different FPGA implementation of the GMM algorithm. In [27], the design of an automated digital surveillance system running in real-time on an embedded platform is presented. The circuit proposed in [27] is able to process 25 fps with frame size 320 × 240 and hence reaches 1.9 Mps.

In [28], the research group of [27], proposes a circuit with improved processing capabilities. When implemented on VirtexII FPGA platform, it processes 1,024 × 1,024 images at 38 fps. Processing capabilities of Jiang et al. [28] are, therefore, 39.8 Mps.

The algorithm of Jiang et al. [29] improves the memory throughput with respect to [28] employing a memory reduction scheme. This entails an increase of the hardware resource utilization. The resulting processing capabilities are, however, reduced with respect to [28] and only reach 7.68 Mps.

In [30], an architecture for a GMM-based classifier based on distributed arithmetic is developed. The implementation is carried out on the Celoxica-RC1000 board equipped with the Xilinx XCV2000E FPGA. The maximum working frequency of the circuit proposed in [30] is 27.01 MHz. It is able to process 27 Mps. Kristensen et al. [27], Jiang et al. [28, 29] and Minghua and Bermak [30] are not able to process 41.5 Mps and, unfortunately, they do not provide information regarding pipeline levels or power dissipation of the systems. Further, the implementations of [20, 23–30] do not comply with the OpenCV GMM algorithm.

In [31], an hardware implementation of the GMM algorithm that complies with the OpenCV algorithm and processes HD videos has been proposed. The circuit of [31] exploits fixed width arithmetic and minimizes the data bandwidth towards the memory.

Additional *Fg/Bg* identification algorithms are proposed in [32–36]. As example, in [35], a hierarchical method for foreground detection that is able to remove the shadows is proposed. Being block based, the algorithm of Guo et al. [35] can trade off processing need with the quality of the detection. When implemented on 2.4 GHz Intel I7 platform, the proposed algorithm is able to process from a minimum of 2.3 Mps to a maximum of 6.8 Mps.

Barnich and Van Droogenbroeck [36] proposes the ViBe algorithm. It is based on integer calculation and hence does not need a floating point unit and provides a random updating of the background that makes the algorithm adaptable to various situations. ViBe does not rely on a particular pdf distribution for the background model and also provides a technique for the fast adaptation of sudden changes of the background. Other characteristics are the resilience to camera displacement and the different adaption time for foreground and ghost images. ViBe has been tested on an Intel 2.67 GHz I7 CPU which provides a processing speed of 61.4 Mps and has also been implemented on a low speed ARM processor with a reasonable processing speed of 0.46 Mps.

Table 1 summarizes the performance of the previously proposed foreground identification processors. In this paper, it has been chosen to focus on the implementation of an FPGA-based processor that implements the GMM algorithm since this is the algorithm proposed in the OpenCV library. As a consequence, Kyrkou and Theocharides [32], Aguilar-Ponce et al. [33], Juvonen et al. [34], Guo et al. [35] and Barnich and Van Droogenbroeck [36] are not considered as a comparison in this paper.

The binary mask provided by the segmentation circuit is usually noisy with *Fg* objects that are often split into several parts. The second step of the segmentation procedure, the denoising phase, processes the binary mask of the *Fg/Bg* tags employing morphological operators with the target of removing the noise and generating solid and separated blocks for the moving objects.

The mathematical morphology [37, 38] is derived from set theory and provides powerful tools for geometrical image analysis, image and video compression, error correction, noise suppression and video segmentation.

Morphology operators can be employed to process binary, gray-scale and color images [39–41]. The architectures that implement morphological operators can be divided into two large classes: systolic arrays [42, 43]; pipelined systems [44].

This paper proposes a hardware system that carries out the phases of identification and denoising. The hardware is composed of the identification block that implements the GMM algorithm and a morphological unit that implements opening and closing operations. Compared with the previously reported contributions, the proposed circuit is able to process HD (1,920 $\times$ 1,080 frame size) video streams in real time when implemented on FPGA circuits, while reducing the programmable logic occupation. Further, the proposed circuit is compatible with the OpenCV implementation of the GMM algorithm, a feature that improves both the accuracy of the system and makes the circuit useful for a wide base of OpenCV developers. The circuit is implemented on different FPGA devices varying the number of pipeline levels. In every case, the programmable logic occupation, the speed and the dynamic power dissipation of the implementation are reported. As example, when implemented on xc5vlx50 Virtex5 FPGA, the proposed circuit works at 50.5 MHz. Programmable logic occupation and power dissipation are 4.09% of the FPGA and 0.46 mW/MHz. Implemented on StratixII, the circuit can process HD videos at 21 fps using 5,038 Logic Elements, 7,957 flip flop and exhibiting a dynamic power dissipation of 4.03 mW/MHz.

The paper is organized as follows. In Sects. 2 and 3 the GMM algorithm and the Morphological operation are described. The proposed hardware implementation and its performance are detailed in Sects. 4 and 15.

2 Gaussian mixture model

The GMM algorithm [20] is a probabilistic algorithm particularly suited to identify moving objects in multimodal backgrounds in which shadows and objects showing

Table 1 Synthetic characteristic for a selection of previously proposed foreground detection algorithms and GMM processors

Ref. no.	Performance (Mps)	Technology	Bandwidth (GB/s)	Energy (mW/MHz)	GMM based	OpenCV compatible
Ref. [20]	0.2	CPU	N.A.	N.A.	Yes	No
Ref. [24]	2.2	CPU (Xeon 5150)	N.A.	N.A.	Yes	No
Ref. [25]	N.A.	N.A.	Higher than GMM	N.A.	Yes	No
Ref. [26]	N.A.	CPU (I7 2.8 GHz)	N.A.	N.A.	Yes	No
Ref. [27]	1.9	FPGA	N.A.	N.A.	Yes	No
Ref. [28]	39.8	FPGA	N.A.	N.A.	Yes	No
Ref. [29]	7.68	FPGA	0.17	N.A.	Yes	No
Ref. [30]	27.0	FPGA	N.A.	N.A.	Yes	No
Ref. [31]	47.0	FPGA	1.08	0.59	Yes	Yes
Ref. [35]	2.3–6.8	CPU (I7 2.4 GHz)	N.A.	N.A.	No	No
Ref. [36](a)	61.4	CPU (I7 2.67 GHz)	N.A.	N.A.	No	No
Ref. [36](b)	0.46	ARM	N.A.	N.A.	No	No

Additional info regarding [31] is in Table 2

N.A. not available

repetitive motion (e.g. rippling waves or wishing leaves) are present. The statistic of each pixel is modeled with a mixture of K Gaussian distributions. Each Gaussian has a different mean variance and a different weight. The sum of the K Gaussian distribution represents the probability distribution of the intensity of a given pixel. In this way, a pixel exhibiting a fixed intensity will be represented with a Gaussian with weight equal to one centered at the pixel intensity and with very small variance, while the remaining $K-$ 1 Gaussian distribution will have weight equal zero. A pixel whose intensity oscillates between two values is perfectly modelled with two Gaussians, centered on the two values and whose weight is 0.5. The number of Gaussian distributions is fixed and equal for each pixel of the frame.

The proposed implementation employs three Gaussian distributions to process grayscale 8 bit per pixel HD images.

2.1 Statistical model

Each Gaussian is represented by its weight (w), by the mean (μ) and by the variance (σ^2). These parameters differ for each Gaussian of each pixel and change for each frame of the video sequence. Their values must be updated to adapt the background model to the changes of the scene.

In the following, the Gaussian parameters for each pixel are defined by two indexes: k is the index for the Gaussian distribution and t is the index that refers to the time instant of the considered frame.

2.2 Parameters update

The following procedure is performed for each pixel of a new frame:

1. The K Gaussian distributions are sorted according to a priority parameter named Fitness (F) and given by:
$$F_{k,t} = w_{k,t}/\sigma_{k,t} \tag{1}$$
The fitness value is a measure of how much a Gaussian distribution is representative of the background and is based on the fact the ideal Gaussian distribution representing a background pixel with a fixed intensity will have large weight and very small variance.
2. A match condition is verified between the pixel value and the K Gaussian distributions that model its statistics. The match condition is:
$$M_k = 1 \quad if \mid (pixel - \mu_{k,t}) \mid < 2,5\sigma_{k,t} \tag{2}$$
A pixel can verify (2) for more than one Gaussian. The Gaussian that matches the pixel ($M_k = 1$) and has the highest F value is considered as the 'matched distribution'.
3. The parameters of the 'matched distribution' are updated as follows:
$$\begin{aligned} &\mu_{k,t+1} = \mu_{k,t} + \alpha_{k,t} \cdot (pixel - \mu_{k,t}) \\ &w_{k,t+1} = w_{k,t} - \alpha_w \cdot w_{k,t} + \alpha_w \\ &\sigma^2_{k,t+1} = \sigma^2_{k,t} + \alpha_{k,t} \cdot [(pixel - \mu_{k,t})^2 - \sigma^2_{k,t}] \\ &\alpha_{k,t} = \alpha_w / w_{k,t} \\ &matchsum_{k,t+1} = matchsum_{k,t} + 1 \end{aligned} \tag{3}$$
where *matchsum* is a counter introduced in the OpenCV algorithm [22] to obtain a faster initial learning phase with respect to the algorithm of Stauffer and Grimson [20]. α_w and $\alpha_{k,t}$ are the learning rates for the weight and for the mean and the variance, respectively.

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

$$w_{k,t+1} = w_{k,t} - \alpha_w \cdot w_{k,t} \tag{4}$$

If no Gaussians verify condition (2), a specific 'no_ match' updating procedure is executed. The parameters of the Gaussian with smallest F value are updated as:

$$\begin{aligned} \mu_{k,t+1} &= pixel \\ w_{k,t+1} &= 1/(matchsum\_tot + 1) \\ \sigma^2_{k,t+1} &= variance\_init \\ matchsum_{k,t+1} &= 1 \end{aligned} \tag{5}$$

where *variance_init* is a fixed initialization value and *matchsum_tot* is the sum of the *matchsum* of the K-1 Gaussians with highest F. The weights of the other K-1 Gaussians are decremented as in (4); means and variances are unchanged.

2.3 Background identification

The Bg identification is performed adding in succession the weights of the first b Gaussian distributions beginning from the one with highest F value, until their sum is greater than a fixed threshold (T) belonging to the (0,1) interval:

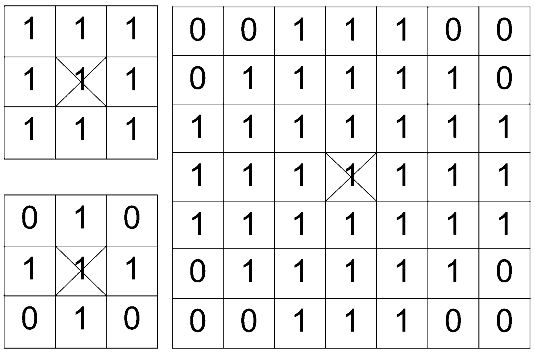
$$B = \arg\min_b \left(\sum_{k=1}^{b} w_{k,t} > T \right) \tag{6}$$

The set of Gaussian distributions that verify (6) represents the Bg and a pixel that matches one of these distributions is classified as a Bg pixel.

3 Morphology

Mathematical morphology [37, 38] provides a wide range of operators for the image processing, all based around the set theory. The common usage includes edge detection, noise removal, image reconstruction, and image segmentation. The operators are particularly useful for the analysis of binary images but they have been extended and applied to gray scale [39, 40], and color images [41].

Erosion and dilation are the two most basic operators. They take as input the image (I) to be eroded or dilated and a structuring element (SE) that determines the effect of the operators on I. The SE can be viewed as a small binary image, represented as a set of pixels on a grid, assuming the values 1 if the pixel belongs to SE and 0 otherwise (Fig. 1). An origin of the SE must be identified. When a morphological operation is carried out, the SE is superimposed on I so that its origin coincides with the input pixel. The origin of the SE is shifted on I, the points within SE are compared with the underlying image and a decision is taken based on the comparison.



1	1	1
1	$\cancel{1}$	1
1	1	1

0	1	0
1	$\cancel{1}$	1
0	1	0

0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
1	1	1	$\cancel{1}$	1	1	1
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0

Fig. 1 Examples of *SE*. The origin (*crossed element*) is in the *center* of the *SE*

The effect of erosion on I is to erode the boundaries of foreground regions. Thus, the area of Fg regions decreases and holes within Fg objects become larger. To compute the erosion, the origin of SE is superimposed on each pixel in I. If, for every pixel in SE, any of the corresponding pixels in I belongs to the Bg, the input pixel is also set to the Bg value, otherwise it remains unchanged.

The effect of the dilation on I is to enlarge the boundaries of foreground objects. Thus, the area of foreground regions grows while the holes within those objects become smaller. To compute the dilation, the origin of SE is superimposed on each pixel in I. If at least one pixel in SE coincides with a Fg pixel in the image underneath, the input pixel is set to the Fg value, otherwise it remains unchanged.

Figure 2 illustrates the effect of the erosion (Fig. 2c) and of the dilation (Fig. 2d) on a binary image (Fig. 2a) with a cross shaped 3×3 SE (Fig. 2b). In Fig. 2, the Fg regions are represented with white pixels (binary value 1), while black pixels (binary value 0) denote Bg. The effect of the erosion on a single Fg pixel surrounded by Bg pixels, is the removal of the Fg pixel. In Fig. 2c, one row and one column are removed from each side of the square of Fig. 2a. The effect of the dilation on a single Fg pixel surrounded by Bg pixels, is the transformation of the Fg pixel in a cross equal to the SE. In Fig. 2d, one row and one column are added on each side of the square of Fig. 2a.

It is worth highlight that erosion and dilation are dual operators and eroding Fg pixels is equivalent to dilating the Bg pixels. Dilation and erosion operators are indicated with the $\oplus$ and $\ominus$, respectively. The combination of dilation and erosion forms other morphological operations such as the opening, denoted with $\circ$, and the closing, denoted with $\bullet$:

$$\begin{aligned} I \circ SE &= (I \ominus SE) \oplus SE \\ I \bullet SE &= (I \oplus SE) \ominus SE \end{aligned} \tag{7}$$

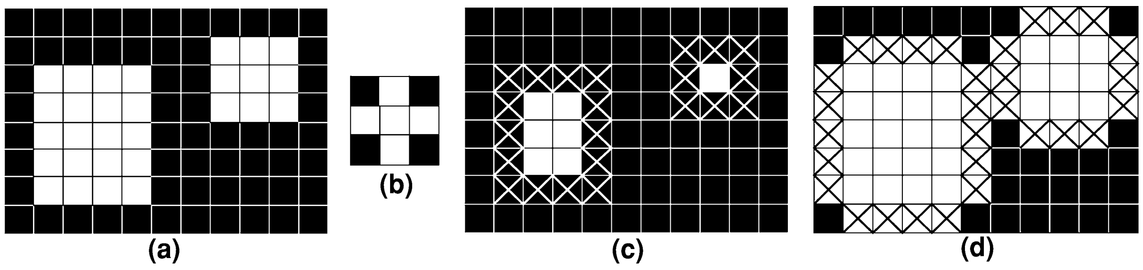


Fig. 2 **a** Original *Fg* mask; **b** structuring element with a cross shape in which the origin element is at the center of the cross; **c** eroded mask; **d** dilated mask. *Crossed pixels* in **c** are the eroded pixels. *Crossed pixels* in **d** are the dilated pixels

4 Hardware implementation

The block diagram of the proposed circuit is shown in Fig. 3. The architecture is divided into the identification block that implements the OpenCV version of the GMM, and the morphological unit, that implements erosion, dilation, opening and closing operations.

The proposed HW design is described in VHDL code and has been accurately optimized to achieve real time processing of HD (1,920 × 1,080 frame size) video sequences. The target devices for the implementations are commercial FPGA devices. The identification block implements the OpenCV version of the GMM algorithm making it useful for a wide base of designers. The performances of the identification block, optimized for the FPGA implementation, to the best of our knowledge, overcome the performance of previously proposed circuits. The morphological unit allows the transparent denoising operation. The proposed implementation of the morphological unit provides useful information regarding the feasibility of the circuit and its performance when implemented on the most commonly available FPGA devices. Additional feature of the proposed circuit is the reduced bandwidth toward the external memory. To reduce the memory bandwidth, the proposed circuit employs an optimized fixed point representation instead of the floating point one used in the OpenCV algorithm. Figure 4 shows that the difference (Fig. 4c) between the *Fg* mask obtained with a software floating point implementation of the OpenCV algorithm (Fig. 4a) and the *Fg* mask obtained with the fixed point hardware implementation (Fig. 4b) is very small. The percentage of differently classified pixels between Fig. 4a, b, for the last frame shown in Fig. 4, is 3%. The use of fixed point representation on a reduced number of bits has also allowed the design of the proposed high performance arithmetic circuit through the

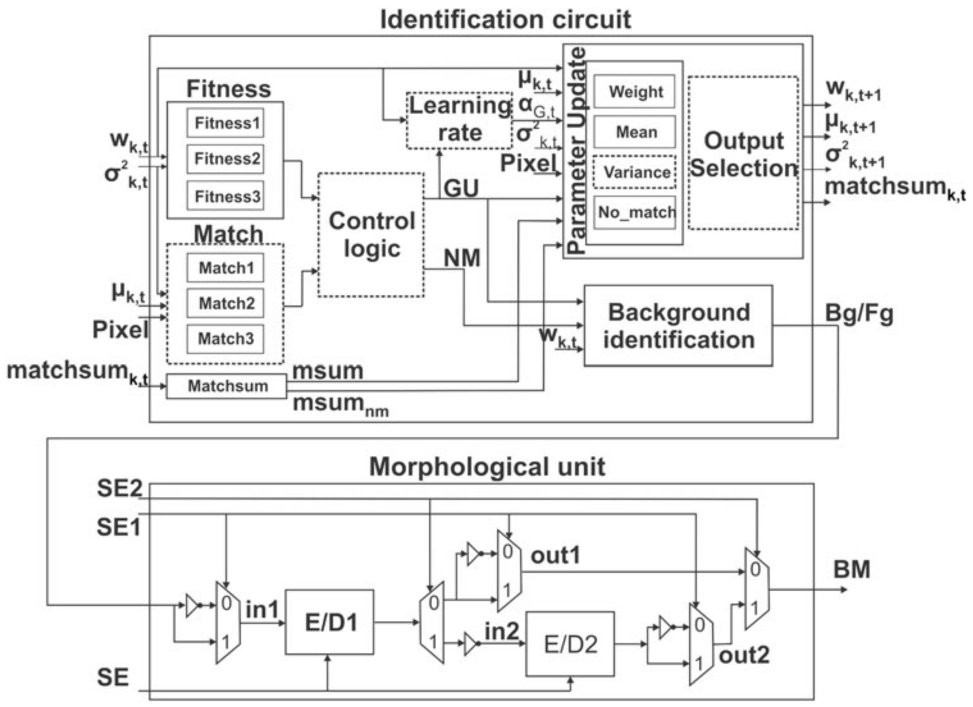


Fig. 3 Block diagram. The identification circuit implements the GMM algorithm while the morphological unit implements erosion, dilation, opening and closing operations

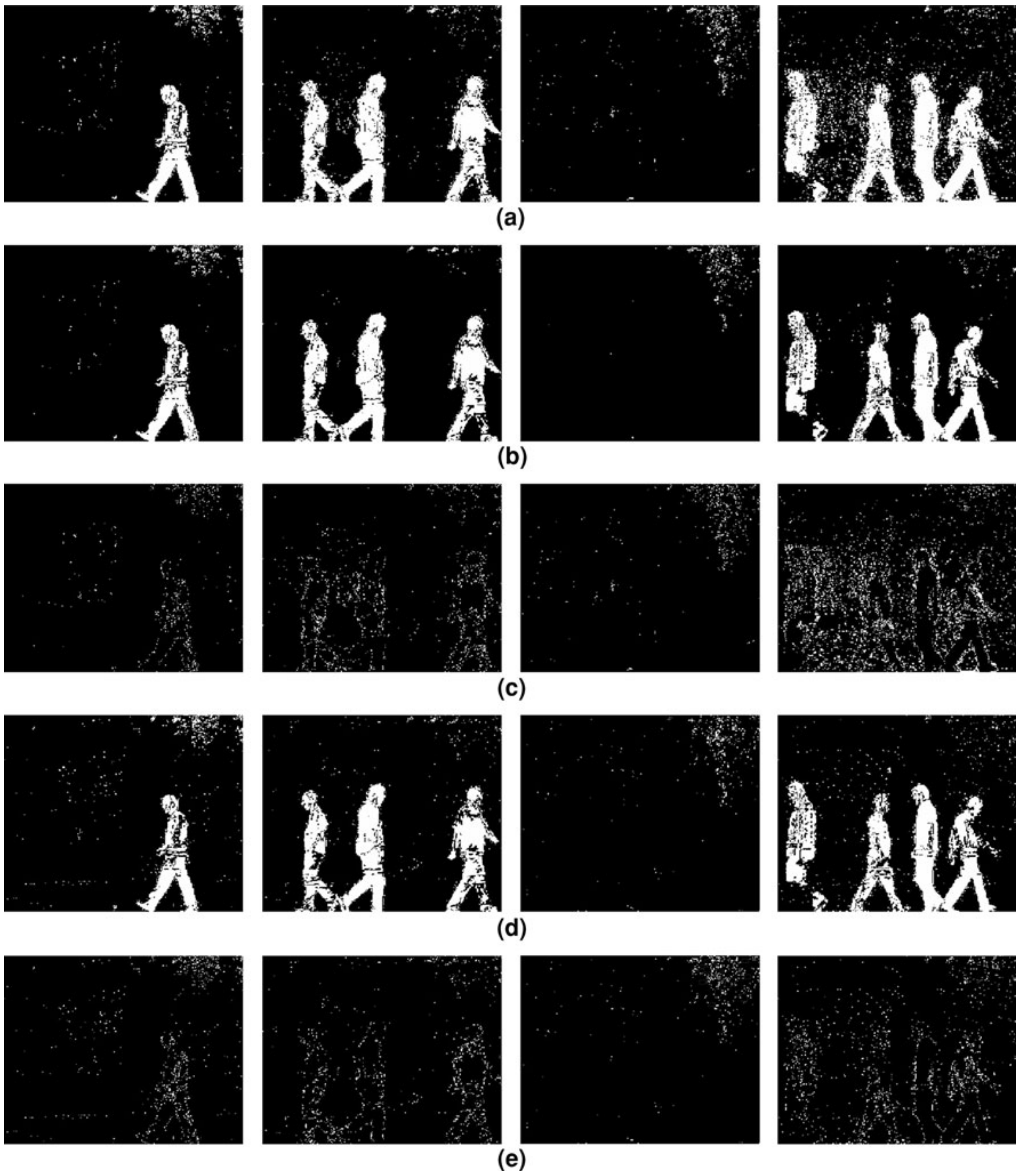


Fig. 4 Frames taken from processed video sequences. **a** The result of the original OpenCV algorithm; **b** the result of the HW implementation that uses the binary multipliers; **c** the difference between **a** and **b** frames; **d** the result of the HW implementation that uses the shifters; **e** the difference between **b** and **d** frames

implementation of non linear arithmetic operations with a ROM-based approach.

4.1 Identification circuit

The input data of the identification circuit are the 8 bit luminance of the input pixel (*Pixel*), and the statistical model of the pixel for the given frame ($\mu_{k,t}$, $\sigma^2_{k,t}$, $w_{k,t}$, $matchsum_{k,t}$, $k = [1,2,3]$) while the outputs are the updated statistical model ($\mu_{k,t+1}$, $\sigma^2_{k,t+1}$, $w_{k,t+1}$, $matchsum_{k,t+1}$) and the *Bg/Fg* tag for the input pixel. The identification circuit implements the equations reported in Sect. 2. In the following, a description of the circuital blocks that compose the identification circuit is given.

Fitness Computes the Fitness factor using (1) for the three Gaussian distributions. Equation (1) requires both a binary inversion and a square root operation. The proposed implementation uses a ROM, in which the pre-calculated outputs for the inverse of the square root of the variance input are stored, and a multiplier, that multiplies the ROM output by the weight of the Gaussian. Since the variance and the inverse of the standard deviation are represented on 12 bit and 8 bit, respectively, ROM size is 212 × 8 bit. The logic utilization on Virtex5 xc5vlx50 is 132 FPGA slices. Maximum combinatorial delay and power dissipation are 6.41 ns and 0.05 mW/MHz, respectively. For a target speed of 41.5 Mps, the allowed combinatorial delay of the circuit is 24 ns and hence the Fitness block accounts for 26.6% of the maximum circuit delay.

Match Verifies the match condition of (2). The implementation is:

$$M_k = 1 \quad if\,(pixel - \mu_{k,t})^2 < 6.25\sigma_{k,t}^2 \tag{8}$$

The circuit requires a subtractor, a squarer, a multiplier, and a comparator. Its logic utilization is 30 LUT and 2 DSP blocks when implemented on xc5vlx50 Virtex5 FPGA. The maximum combinatorial delay is 6.88 ns that accounts for 28.6% of the maximum allowed circuit delay.

Matchsum The $matchsum_{k,t}$ signal is a counter associated to each Gaussian that has been introduced in the OpenCV algorithm. The Matchsum block increments the $matchsum_{k,t}$ of the matched Gaussian. In [20], the $matchsum_{k,t}$ is not used and, when no match is verified, the weight is initialized to a fixed (very small) value. Using the $matchsum_{k,t}$ value the OpenCV GMM method allows a fast identification of the initial *Bg* of the scene [31].

Control logic Sorts the Gaussians according to *F* value and establishes which Gaussian is updated as in (3), (4), (5).

Learning rate computes the learning rate $\alpha_{k,t}$ as in (3). Updates mean and variance of the matched distribution.

The implementation of (3) and (4) implies the use of multipliers. In the proposed implementation, the α_w and $\alpha_{k,t}$ values are quantized as power of two ($\alpha_w = 2^{nw}$ and $\alpha_{k,t} = 2^{nk}$), allowing the replacement of the multipliers with shifters. The *nw* value is hardwired while the *nk* values that better approximate $\alpha_{k,t}$ as a function of $w_{k,t}$ are stored in a small ROM that uses few LUTs of the target FPGA. The quantization of the learning rate introduces a small approximation on the GMM algorithm. The result of the processing of a sample video using the shifters is shown in Fig. 4d. The difference with the result of the processing using the multipliers is shown in in Fig. 4e. It is possible to note that the output of the circuit is very similar to the *Bg* mask obtained with the use of multipliers. The percentage of differently classified pixels is 0.4%.

Background identification Checks condition (6) calculating the *Bg/Fg* tag.

Parameter update and output selection Weight, mean, and variance blocks of the parameter update unit, update the parameters according to (3), (4) when the match condition is verified. If no Gaussian matches the pixel, the No_match block updates mean, variance and weight of the Gaussian with smallest *F* value according to (5). The parameter update unit uses *nk* and *nw* values to perform the shift operations instead of multiplications. The output selection unit establishes the values of the updated parameters depending on whether the match condition is verified or not.

The proposed GMM block improves the hardware implementation of the single channel, luminance based, OpenCV version of the GMM algorithm proposed in [31]. In [31], the GMM has been implemented using three Gaussians per pixel and processes HD video on xc5vlx50 Virtex5 FPGA. The circuit proposed in [31] processes 22 fps with a dynamic power dissipation equal to 27.6 mW@47MHz and using 5.5% of the available LUT. The implementation proposed in this paper, on the same target FPGA and with the same number of Gaussians per pixel, is able to process 24 fps, reduces the power dissipation up to 0.46 mW/MHz that corresponds to 21.62 mW@47MHz, and uses 3.70% of the Slice LUT in the FPGA. The comparison of the performances together with the percentage improvement on each value, are reported in Table 2. Table 2 refers to the identification unit of the proposed circuit (see Fig. 3).

The improvement in performance with respect to [31] is mainly due to the substitution of some of the multipliers present in [31] with shifters. This only requires the approximation of certain circuit internal variables to the nearest power of two. The worsening in the accuracy of the circuit has been tested comparing videos processed with both circuits. The results of Fig. 4 show that the worsening in accuracy is negligible and can be surely recovered through the morphological unit. As example, in a sample video composed of 300 frames (272 × 176), the average number of differently classified pixels per frame is 186 that corresponds to the 0.4% of the pixels.

4.2 Morphological unit

The morphological unit of Fig. 3 implements erosion, dilation, opening and closing operations. The input signal of the circuit are the *Bg/Fg* tag, the *SE* and two control signals (*SEL1* and *SEL2*) and provides the *Bg* mask (*BM*) as output. *SEL1* and *SEL2* are two binary signals that establish the operation to be performed according to Table 3. Opening and closing can be derived by erosion and dilation as in (7). Moreover, the erosion and dilation operations are one the dual of the other. It is, therefore, possible to obtain the erosion of the image *I* using the dilation operator if the input image *I* and the result of the dilation are inverted. As a consequence, all the considered morphological operators can be implemented using only the dilation operator.

Table 2 Comparison between the proposed identification circuit and Ref. [31]

Virtex5 implementation	# LUT	Energy (mW@47MHz)	Frame rate (fps)
Ref. [31]	1,572	27.60	22
Proposed identification circuit	1,066 (−32%)	21.62 (−22%)	24 (+9%)

Table 3 Operators performed by the morphological unit of Fig. 3 as a function of the signals SEL1 and SEL2

SEL1	SEL2	Operation	In1
0	0	Erosion	Not (*Bg/Fg*)
1	0	Dilation	*Bg/Fg*
0	1	Opening	Not (*Bg/Fg*)
1	1	Closing	*Bg/Fg*

$$
\begin{aligned}
&I \ominus SE = (I' \oplus SE)' \\
&I \circ SE = (I' \oplus SE)' \oplus SE \qquad (9)\\
&I \bullet SE = ((I \oplus SE)' \oplus SE)'
\end{aligned}
$$

E/D is the circuital block that implements the dilation operation. It is duplicated, as shown in Fig. 3, in order to allow closing and opening operations. Depending on the *SEL1* value, *Bg/Fg* tag or its binary inversion are sent as input to the E/D unit (Table 3). The *Bg/Fg* tag is a binary value: '0' if the pixel has been classified as *Bg*; '1' otherwise.

The E/D unit is based on a delay line architecture. The input binary values *Fg/Bg* are stored in some shift registers as shown in Fig. 5. If *SE* size is equal to $m\times n$ and *I* size is $i \times j$, *m* shift registers are employed. (1 shift register *n* bit long; $m - 1$ shift registers *j* bit long). When all the shift registers are full (Fig. 5b), as an incoming pixel is shifted in, the oldest pixel is shifted out. The pixels covered by the *SE* are stored using flip flops while the other pixels are stored in FIFOs (Fig. 6). In order to reduce the register utilization, when the circuit is implemented on Virtex or Spartan FPGA, the FIFOs are implemented with LUT (Table 4) instead of flip flops. The dilation is performed on the pixels stored in the flip flops using the logic shown in Fig. 6.

One E/D unit is needed to perform erosion and dilation. Opening and closing require the implementation of a second unit to process the output of the first one, as highlighted in (9). The implementation of the morphological operators at the boundary of the figure is different from those in the interior of the frame (Fig. 7a). Usually, two solutions are adopted. The first one is to add extra control logic to avoid the boundary pixel processing. The second one is the padding technique, Fig. 7b that increases the frame size. Additional pixel are inserted outside the boundary of *I*. Since they should not affect the result of the operations, the padding area is filled with '1' for the erosion and '0' for the dilation. Padding increments the output delay and, more important, fragments the dataflow. In this paper, a technique that combines the two methods is adopted. It allows the processing of the boundary pixels without stopping the dataflow. In the proposed technique, a control logic is activated when a boundary pixel has to be processed and fixes some nodes, originally connected to the output of registers, at logic value '0'. The required logic consist of two counters and some comparators to identify the boundary pixel. This entails an increase of circuit area but does not increase the output delay that can be very high when HD videos are processed. The same counters are used to synchronize the two E/D units when opening and closing are performed. Figure 8 shows frames on which Erosion, Dilation, Opening and Closing operations have been performed.

5 Results and performances

The proposed circuit has been implemented on Virtex5, VirtexII, Virtex4 and Spartan3 Xilinx FPGA and on StratixII and CycloneII Altera FPGA. The synthesis for Virtex5, Virtex4 and Spartan3 has been conducted using Synplify while XST has been used for VirtexII synthesis. Fitting and P&R have been conducted using ISE in all the

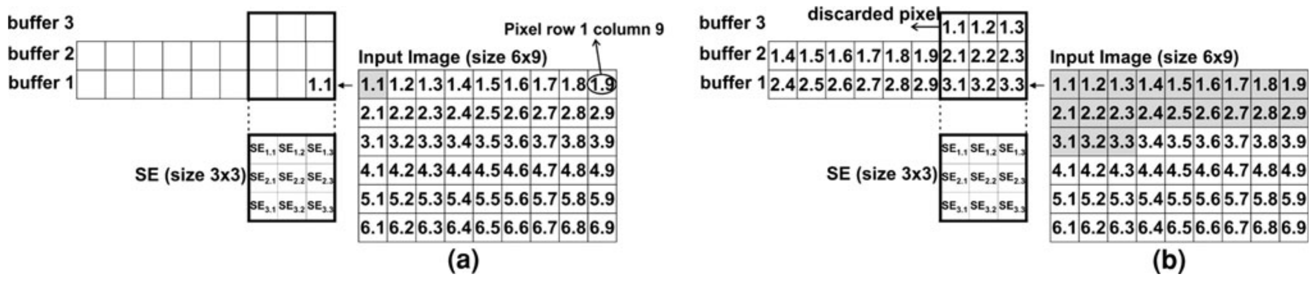


Fig. 5 Example of delay lines used to perform morphological operations. In **a**, the input image is shifted into three buffers (*SE* size 3 × 3) whose width depends on the image size (6 × 9). When the buffers are full (**b**), the first pixel is discarded

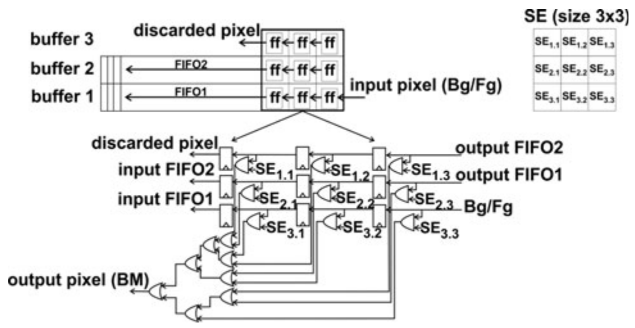


Fig. 6 Delay lines implementation with a 3 × 3 *SE*

cases. The synthesis for StratixII and CycloneII has been conducted using QuartusII. Circuit simulations use ModelSim XE and ModelSim Altera for Xilinx and Altera FPGA, respectively. Both simulation tools also generate the 'vcd' files necessary for the accurate determination of the power dissipation. Power dissipation has then been computed using XPower Analyzer software (Xilinx) and PowerPlay Power Analyzer tool (Altera).

The proposed circuit, for every pixel that is processed, fetches from the memory 108 bit for the background model and 8 bit for the pixel value. Afterward, the circuit stores the updated background model, composed of 108 bit, into the memory. The total number of bit transferred, for each pixel, is therefore 224 (28 bytes). Since in the considered applications the frame rate is always 20 fps (independently on the maximum working frequency of the background identification circuit) this implies that the processed pixels per second are 41.5×10^6 (20 fps multiplied by 1,920 × 1,080) each one requiring 28 bytes of data. Hence the proposed circuit features a 1.08 GB/s bandwidth toward the memory (calculated as $41.5 \times 10^6 * 28/1{,}024^3$).

The analyses have been conducted including input and output registers that synchronize the circuit and provide timing performances that are not dependent on the I/O pads. The circuit has been tested using artificial videos, computer animated videos with simple backgrounds and through video sequences taken from real surveillance cameras. The circuit performs optimally and run smoothly without showing reliability problems.

In order to fairly compare the proposed circuit, an analysis of the performances as a function of the number of pipeline levels present in the identification part of the circuit (see Fig. 3) has also been carried out. The placing and the number of pipeline registers have been optimized using the retiming feature of Synplify synthesizer.

Table 5 shows the performances of the proposed circuit as a function of the target FPGA, of the number of pipeline levels and of the frame size. The bandwidth toward the memory for the circuits reported in Table 5, assuming that the circuit process the target HD video stream of 20 fps, is always 1.08 GB/s since the required bit of data for each pixel is fixed to 224. If a different frame rate or a different image resolution is needed, the bandwidth scales proportionally. As example if a video stream with 320 × 240 image resolution is processed at 10 fps, the required bandwidth toward the memory is 20.5 MB/s.

5.1 Virtex5 xc5vlx50 speed grade-3 implementations

As shown in Table 5, the proposed circuit implemented on Virtex5 FPGA without pipeline has a maximum working frequency of 50.5 MHz when implemented to process both 1,920 × 1,080 frames. It uses 1,179 Slice LUT that is 4.09% of FPGA logic resources and 236 LUT are used as memory to implement the shift registers of the morphological unit. The power consumption is 0.46 mW/MHz. With one level of pipeline, the number of Slice register increases up to 492, and the number of Slice LUT and the power dissipation are 1,182 and 0.22 mW/MHz, respectively.

5.2 VirtexII xc2v1000 implementation

In [28], a *Fg/Bg* and denoising architecture, implemented on VirtexII xc2v1000 FPGA, is proposed. The maximum clock frequency is 40 MHz that allows the circuit to process 38 fps with 1024 × 1024 frame size. No information regarding the number of pipeline level of the architecture is given.

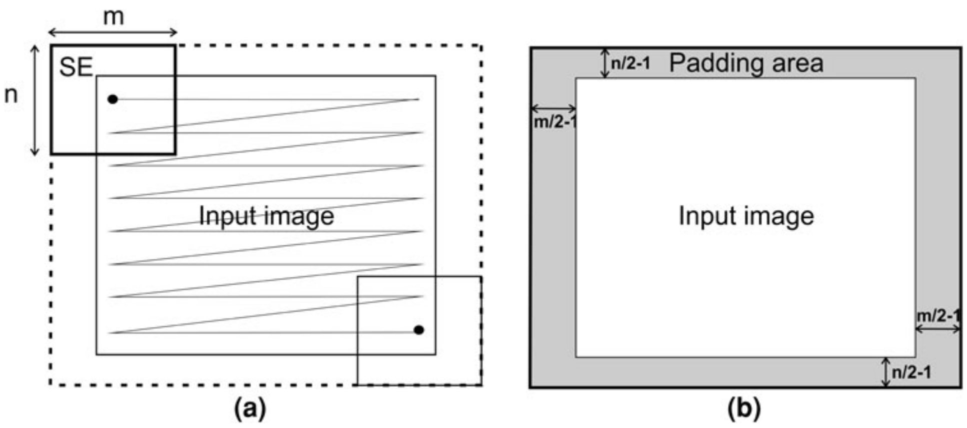
As shown in Table 5, when the identification circuit is implemented with one level of pipeline, the proposed circuit outperforms the circuit of [28]. The proposed circuit has a maximum working frequency of 50.1 MHz; uses 1646 LUT and 460 Slice registers. The dynamic power dissipation is 4.90 mW/MHz.

Table 4 Performances of morphology unit implemented on Virtex5 FPGA

Circuit	FIFO implementation	Frequency (MHz)	Slice register	Slice LUT	Slice
E/D	Flip flop	374.25	1.946	227	542
E/D	LUT	432.71	30	173	51
Morphology unit	Flip flop	263.37	3,898	494	1,078
Morphology unit	LUT	341.53	56	361	106

Frame size 1,920 × 1,080

Fig. 7 **a** Boundary problem. *SE* stretches outside the image borders. **b** Frame padding necessary for an *SE* whose size is $m \times n$



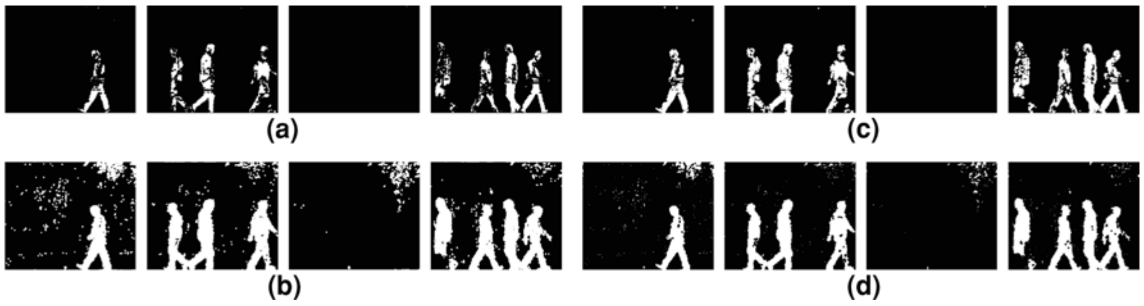


Fig. 8 Results of morphological operators on the frames shown in Fig. 4. The frames are the output of the identification circuit of Fig. (3). **a** Erosion, **b** dilation, **c** opening, **d** closing

Table 5 Performances of circuit of Fig. 3

FPGA (Xilinx)	Pipe levels	Frame size	Frequency (MHz)	# Slice LUT employed as:		# Flip flop	# Slice	# DSP-MULT	Dynamic power (mW/MHz)	fps
				Logic	Memory					
Virtex5 xc5vlx50	0	1,920 × 1,080	50.5	943/28,800	236/28,800	291/28,800	452/14,400	10	0.46	24
	1	1,920 × 1,080	90.9	942/28,800	240/28,800	492/28,800	476/14,400	10	0.22	43
VirtexII xc2v1000	1	1,024 × 1,024	50.1	1,390/10,240	256/10,240	460/10,240	1121/5,120	10	4.90	47
Virtex4 xc4vfx12	0	1,920 × 1,080	38.9	1,348/10,944	900/10,944	291/10,944	1264/5,472	10	1.38	18
Spartan3 xc3s1000	0	1,920 × 1,080	22.3	1,412/15,360	900/15,360	291/15,360	1253/7,680	7	0.89	10
	1	1,920 × 1,080	37.2	1,457/15,360	900/15,360	476/15,360	1347/7,680	7	0.41	17

FPGA (Altera)	Pipe levels	Frame size	Frequency (MHz)	# Logic element	# Flip flop	# DSP	Dynamic power (mW/MHz)	fps
StratixII EP2S15F484C3	0	1,920 × 1,080	44.03	5,038/12,480	7,957/12,480	11	4.03	21
CycloneII EP2C15AF484C6	0	1,920 × 1,080	30.47	9,192/14,448	7,957/14,448	11	1.61	14

5.3 Virtex4 xc4vfx12 implementation

Not pipelined circuit implemented on Virtex4 xc4vfx12 FPGA allows to process 18 fps with 1,920 × 1,080 frame size. The resulting implementation uses 2,248 Slice LUT and 291 Slice registers. The dynamic power dissipation is 1.38 mW/MHz.

5.4 StratixII EP2S15F484C3 implementation

Maximum frequency is 44.03 MHz, faster than the circuit implemented on Virtex4 but slower than the one implemented on Virtex5. The logic utilization is equal to 40.37 % of the total number of Logic Elements and 7,957 of 12,480 flip flop are employed. The power dissipation is

4.03 mW/MHz and it is comparable with the power dissipation of the Virtex2 implementation.

5.5 Spartan3 xc3s1000 implementation

The proposed circuit has been implemented on Spartan3 with zero and one levels of pipeline. The maximum clock frequency and the number of frame per second that the circuit can process with frame size 1,920 × 1,080 and with one pipeline level are 37.2 MHz and 17 fps, respectively. Area utilization is 15.35% of total number of LUT and 3.10% of the total number of Slice registers.

5.6 CycloneII EP2C15AF484C6 implementation

The performances on Cyclone EP2C15AF484C6 FPGA are 9,192 Logic Elements equal to 63.6% of total Logic Elements of the FPGA, 7,957 Flip Flop, dynamic power dissipation is 1.61 mW/MHz. The implemented circuit, without pipeline levels, provides a maximum frequency higher than not pipelined low cost Xilinx Spartan3 implementation but also an higher resource utilization.

6 Conclusions

This paper presents an hardware system, implemented on FPGA circuits, to perform segmentation and morphology phases of an identification system. The proposed circuit improves the GMM implementation proposed in [31] and is able to process HD video streams in real time. The GMM algorithm is based on the OpenCV algorithm and has been implemented with particular attention to the reduction of the bandwidth toward the memory and to the optimization of hardware utilization. Thus, the circuit exploits fixed point arithmetic (instead of the double precision floating point one used in the OpenCV) and the multipliers in some of the equations have been replaced with shifters. The morphological unit is based on a delay line architecture where, whenever possible, shift registers have been implemented with LUT to process HD videos with a reduced resource utilization.

The performances of the architecture on different FPGA are provided. The use of pipeline levels is also considered as a method to optimize circuit performances.

The circuit implemented on Virtex5 (xc5vlx50) without levels of pipeline allows a maximum working frequency of 50.5 MHz and is hence able to process 24 fps for an HD video with frame size 1,920 × 1,080. The logic utilization is 1,179 slice LUT (4.09% of the available LUT) while the dynamic power dissipation is 0.46 mW/MHz.

References

1. Gutchess, D., Trajković, M., Cohen-Solal, E., Lyons, D., Jain, A. K.: A background model initialization algorithm for video surveillance. In: Proceedings of the Eighth IEEE International Conference on Computer Vision, Vancouver BC, July 2001, vol.1, pp. 733–740
2. Haritaoglu, I., Harwood, D., Davis, L. S.: A fast background scene modeling and maintenance for outdoor surveillance. In: Proceedings of the 15th International Conference on Pattern Recognition, Barcelona, Spain, 2000, vol.4, pp. 179–183
3. Haritaoglu, I., Harwood D., Davis, L.S.: W4: real-time surveillance of people and their activities. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 809–830 (2000)
4. KaewTrakulPong, P., Bowden, R.: A real time adaptive visual surveillance system for tracking low-resolution color targets in dynamically changing scenes. Image Vis. Comput. **21**(10), 913–929 (2003)
5. Gloyer, B., Aghajan, H. K., Siu, K., Kailath, T.: Video-based freeway-monitoring system using recursive vehicle tracking. In: Proceedings of the SPIE Symposium on Electronic Imaging: Image and Video Processing, San Jose, CA, USA, 1995, pp. 173–180
6. Vibha, L., et al.: Moving vehicle identification using background registration technique for traffic surveillance. In: Proceedings of the International Multi Conference of Engineers and Computer Scientists, Hong Kong, 2008, vol. 1, pp. 572–577
7. Kastrinaki V., Zervakis M., Kalaitzakis K.: A survey of video processing techniques for traffic applications. Image Vis. Comput. **21**(4), 359–381 (2003)
8. Jieménez, H., Salas, J.: Temporal templates for detecting the trajectories of moving vehicles. Lecture Notes in Computer Science, vol. 5807, pp. 485–493 (2009)
9. Porikli, F.: Achieving real-time object detection and tracking under extreme conditions. J. Real-Time Image Process. **1**, 33–40 (2006)
10. Jamil N., Sembok, T.M.T., Bakar, Z.A.: Noise removal and enhancement of binary images using morphological operations. Inf Technol **4**, 1–6 (2008)
11. Chaohui, Z., Xiaohui, D., Shuoyu, X., Zheng, S., Min, L.: An improved moving object detection algorithm based on frame difference and edge detection. In: Proceedings of the Fourth International Conference on Image and Graphics, Chengdu, 2007, pp. 519–523
12. Mingwu, R., Han, S.: A practical method for moving target detection under complex background. Comput. Eng. 33–34 (2005).
13. Benxian, X., Cheng, L., Hao, C., Yanfeng, Y., Rongbao, C.: Moving object detection and recognition based on the frame difference algorithm and moment invariant features. In: 27th Chinese Control Conference, Kunming, 2008, pp. 578–581
14. Chien, S.Y., Ma, S.Y., Chen, L.G.: Efficient moving object segmentation algorithm using background registration technique. IEEE Trans. Circuits Syst. Video Technol. **12**(7), 577–586 (2002)
15. Ridder, C., Munkelt, O., Kirchner, H.: Adaptive background estimation and foreground detection using Kalman filtering. In: Proceedings of the ICRAM, 1995, pp. 193–199
16. Karmann, K.P., Brandt, A.: Moving object recognition using an adaptive background memory. In: Proceedings of Time-Varying Image Processing and Moving Object Recognition, Capellini Ed., 1990, vol.2.
17. Toyama, J., Krumm, J., Brumitt, B., Meyers, B.: Wallflower: Principles and practice of background maintenance. In: International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 255–261

18. Jacques, J.C.S., Jung, C.R., Musse, S.R.: Background subtraction and shadow detection in grayscale video sequences. In: 18th Brazilian Symposium on Computer Graphics and Image Processing, Brazil, 2005, pp.189–196
19. Wren C., Azarbayejani, A., Darrell, T., Pentland, A.: Pfinder: real-time tracking of the human body. IEEE Trans. Pattern Anal. Mach. Intell. **19**(7), 780–785 (1997)
20. Stauffer, C., Grimson, W.: Adaptive background mixture models for realtime tracking. Proc. IEEE conf. Comput. Vis. Pattern Recogn. **2**, 246–252 (1999)
21. Stauffer, C., Grimson, W.: Learning patterns of activity using real-time tracking. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 747–757 (2000)
22. OpenCV library on source forge. http://sourceforge.net/projects/opencvlibrary/
23. Minghua, S., Bermak, A.: An efficient digital VLSI implementation of Gaussian mixture models-based classifier. In: IEEE Transaction on Very Large Scale Integration Systems, 2006, pp. 962–974
24. Varcheie, P., Sills-Lavoie, M., Bilodeau, G.-A.: A multiscale region- based motion detection and background subtraction algorithm. Sensors **10**, 1041–1061 (2010)
25. Lin, H.-H., Chuang, J.-H., Liu, T.-L.: Regularized background adaptation: a novel learning rate control scheme for Gaussian mixture modeling. IEEE Trans. Image Process. **20**(3), 822–836 (2011)
26. Suhr, J.K., Jung, H.G., Li, G., Kim, J.: Mixture of Gaussians-based background subtraction for Bayer-pattern image sequences. IEEE Trans. Circuits Syst. Video Technol. **21**(3), 365–370 (2011)
27. Kristensen, F., Hedberg, H., Jiang, H., Nilsson, P., Öwall, V.: An embedded real-time surveillance system: implementation and evaluation. J. signal Process. Syst. **52**(1), 75–94 (2008)
28. Jiang, H., Ardö, H., Öwall, V. Hardware accelerator design for video segmentation with multi-modal background modelling. In: Proc. ISCAS **2**, 1142–1145 (2005)
29. Jiang H., Ardö H., Öwall V. A Hardware architecture for real-time video segmentation utilizing memory reduction techniques. IEEE Trans. Circuit Syst. Video Technol. **19**, 226–236 (2009)
30. Minghua, S., Bermak, A., Chandrasekaran, S., Amira, A.: An efficient FPGA implementation of Gaussian mixture models-based classifier using distributed arithmetic. In: IEEE International Conference on Electronics, Circuits and Systems, Nice, 2006, pp. 1276–1279
31. Genovese, M., Napoli, E., Petra, N.: OpenCV compatible real time processor for background foreground identification. In International Conference on Microelectronics, Egypt, Cairo, 2010, pp. 467–470
32. Kyrkou, C., Theocharides, T.: A flexible parallel hardware architecture for AdaBoost-based real-time object detection. In: IEEE Trans.Very Large Scale Integration (VLSI) System, Issue: 99, pp 1–14 (2010)
33. Aguilar-Ponce R., et al.: Real-time VLSI architecture for detection of moving object using Wronskian determinant. In: IEEE-MWSCAS, Covington, KY, vol. 1, pp. 875–878 (2005)
34. Juvonen, M.P.T., Coutinho, J.G.F., Luk, W.: Hardware architectures for adaptive background modelling programmable logic. In: SPL, Mar De Plata, 2007, pp. 149–154
35. Guo J.-M., Liu Y.-F., Hsia C.-H., Shih M.-H., Hsu C.-S.: Hierarchical method for foreground detection using codebook model. IEEE Trans. Circuits Syst. Video Technol. **21**(6), 804–815 (2011)
36. Barnich O., Van Droogenbroeck M. (2011) ViBe: A universal background subtraction algorithm for video sequences. IEEE Trans. Image Process. **20**(6), 1709–1724
37. Serra, J.: Image Analysis and Mathematical Morphology. Academic Press Inc., Orlando (1983)
38. Maragos, P.: A representation theory for morphological image and signal processing. IEEE Trans. Pattern Anal. Mach. Intell. **11**(6), 586–599 (1989)
39. Huang, C.T., Mitchell, O.R.: A Euclidean distance transform using grayscale morphology decomposition. IEEE Trans. Pattern Anal. Mach. Intell. **16**(4), 443–448 (1994)
40. Zarandy, A., Stoffels, A., Roska, T., Chua, L.O.: Implementation of binary and grayscale mathematical morphology on the cnn universal machine. IEEE Trans. Circuits Syst. I, Fundam. Theory Appl. **45**(2), 163–168 (1998)
41. Zaharescu, E., Zamfir, M., Vertan, C.: Color morphology-like operators based on color geometric shape characteristics. Int. Symp. Signals Circuits Syst. **1**, 145–148 (2003)
42. Frejes, S., Vajda, F.: A data-driven algorithm and systolic architecture for image morphology. In: Proceedings of the IEEE International Conference on Image Processing, Austin, Texas, vol. 2, pp. 550–554 (1994)
43. Malamas, E.N., Malamos, A.G., Varvarigou, T.A.: Fast implementation of binary morphological operations on hardware-efficient systolic architectures. J. VLSI Signal Process. 25:79–93 (2000)
44. Velten, J., Kummert , A.: FPGA-based implementation of variable sized structuring elements for 2-D binary morphological operations. In: Proceeings of the 1st IEEE International Workshop on Electronic Design, Test and Application Jan 29–31 2002, pp. 309–312

Author Biographies

Genovese Mariangela was born in Italy in 1986. He is a PhD student in Electronic Engineering at the University of Napoli since 2011. He graduated with honors in March 2010 at University of Naples Federico II.

Ettore Napoli was born in Italy in 1971. He is an Associate Professor at the University of Napoli since 2005. He received PhD in Electronic Engineering in 1999. He received MS in Electronic engineering with honors in 1995, Bachelor degree in Physics with honors in 2009. He was a Research Associate at the Engineering Department of the University of Cambridge (UK) in 2004. His scientific interests include VLSI circuit design and modeling and design of power semiconductor devices. In the VLSI field, his interests are high speed arithmetic subsystems and advanced flip-flops. In the power devices field, his main interests are the PiN diode, the vertical IGBT, superjunction devices and Lateral IGBTs. Prof. Napoli is author or co-author of more than 80 papers published in international journals and conferences.