OpenCV compatible real time processor for background foreground identification

M. Genovese, E. Napoli, N. Petra
DIBET – University of Napoli, Via Claudio, 21- 80125 Napoli - Italy
Email: ma.genovese@studenti.unina.it

***Abstract*—The background identification methods are used in many fields like video surveillance and traffic monitoring. In this paper we propose a hardware implementation of the Gaussian Mixture Model algorithm able to perform background identification on HD images. The proposed circuit is based on the OpenCV implementation, particularly suited to improve the initial background learning phase. Bit-width has been optimized in order to reduce hardware complexity and increase working speed. The proposed circuit processes 22 1920X1080 frames per second when implemented on Virtex 5 FPGA.**

***Index Terms* — Background identification, Field Programmable Gate Array, Object detection, OpenCV.**

I. INTRODUCTION

Surveillance and traffic monitoring applications [1]-[4] rely on the identification of relevant events in video sequences. Object detection techniques have been studied during the years and different algorithms have been proposed.

Main real time identification algorithms are based on the frame difference [5]-[7] or on the comparison with a reference model (background subtraction methods) [8]-[13]. The frame difference methods [5]-[7], detect the moving objects comparing consecutive frames. The algorithms are fast and simple to implement but the output depends on the speed of the moving objects.

The background subtraction algorithms detect the foreground comparing the frame with a reference model.

Ref. [8] considers a pixel as background if it keeps a constant value for a relatively long time. The technique fails when a change of the illumination arise on an otherwise constant scene. Papers [9],[10] use a Kalmann filter, while [11] proposes a Wiener filter, to adapt the background model.

Other methods use statistical algorithms to obtain a statistical model for each pixel of the background, [12],[13]. For a new frame, the pixels are compared with the statistical model. If the difference is greater than a threshold, the pixel is classified as foreground. In [12] an algorithm that uses a single Gaussian distribution per pixel, is proposed. The algorithm is efficient but is not able to describe a multimodal background in which shadows and objects showing repetitive motion (e.g. rippling waves or whishing leaves) are present. For this reason, in [13], each pixel is modeled with a mixture of Gaussian distributions. The technique of [13] is known as Gaussian Mixture Model (GMM) and provides good performances in both presence of illumination changes and multimodal background. Due to the good performances the GMM algorithm has been selected as the background detection algorithm in the OpenCV library [15].

OpenCV (Open Source Computer Vision) is a Open Source software library developed by Intel. OpenCV provides a common base of computer vision instruments able to extract relevant details from the images and to process them in automatic way.

The GMM algorithm proposed in the OpenCV library is an optimized version of the algorithm of [13] that is particularly suited to improve the initial learning phase.

Main drawback of the GMM algorithm is the need of numerous non linear computations that linearly increase with the number of Gaussians per pixel and make real time video processing not possible with a software implementation.

In [13], a frame rate of 11fps is obtained for a small frame size of 160x120 on an SGI 02 with a RIOOOO processor. In [14] an FPGA implementation of the GMM algorithm is proposed. The algorithm does not comply with the OpenCV GMM algorithm but improves the processing capabilities and processes relatively large images (1024x1024 at 38 fps).

In this paper a hardware implementation of the GMM algorithm that allows real time processing of high definition videos and complies with the OpenCV, is proposed.

Our circuit processes HD images at 22 fps when implemented on Virtex5 FPGA. Programmable logic occupation is 5.5% of the xc5vlx50 FPGA while power dissipation is 27.6mW@47 MHz.

II. GAUSSIAN MIXTURE MODEL

The GMM algorithm has been proposed by Stauffer and Grimson [13] with the aim of efficiently model a multimodal background. It describes the statistic of each pixel using a statistical model composed by a mixture of K Gaussian distributions. Using more than one Gaussian allows to model real situations. As an example a point on the frame whose intensity oscillates between two values is perfectly modeled with two Gaussians. Greater is the number of the Gaussian distributions employed by the algorithm, higher is the precision of the method and the computational complexity. A description of the GMM algorithm follows. For the details

978-1-4244-5816-5/09/$26.00 ©2009 IEEE

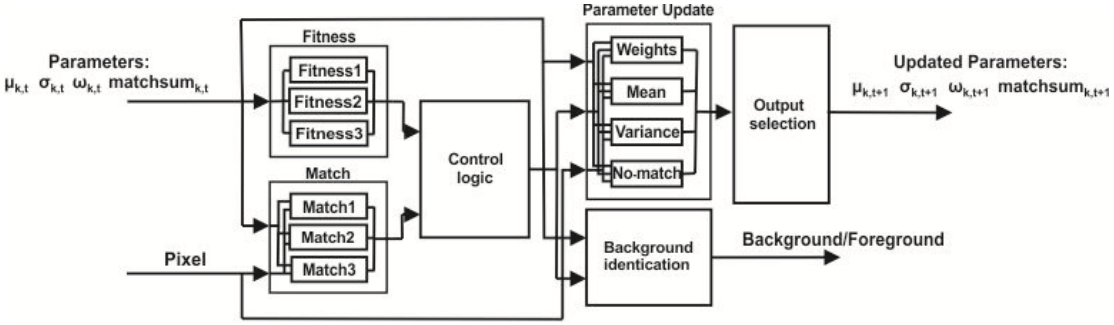

Fig. 1. Block diagram of the background identification circuit

refer to [13] and to the OpenCV documentation.

A. Parameters Update

When a frame is acquired, a match condition with the *K* Gaussians is verified for each pixel. A pixel matches with a Gaussian if the difference between the mean of the distribution and the pixel is lower than 2.5 standard deviations. The match condition is important to establish if the pixel value can be considered as background. When the *k-th* Gaussian verifies this condition (matches), its parameters are updated as follows:

$$\begin{aligned} \mu_{k,t+1} &= \mu_{k,t} + \alpha_\omega \cdot \left(pixel - \mu_{k,t} \right) \\ \sigma^2_{k,t+1} &= \sigma^2_{k,t} + \alpha_\omega \cdot \left[\left(pixel - \mu_{k,t} \right)^2 - \sigma^2_{k,t} \right] \\ \omega_{k,t+1} &= \omega_{k,t} - \alpha_{G,t} \cdot \omega_{k,t} + \alpha_{G,t} \cdot M_k \\ matchsum_{k,t+1} &= matchsum_{k,t} + 1 \end{aligned} \quad (1)$$

where μ, σ^2 and ω are mean, variance and weight for each Gaussian. $\alpha_{G,t}$ and α_ω are two different learning rates and $matchsum$ is a counter introduced in the OpenCV algorithm that will be described in the following. For the unmatched Gaussians the weights are updated according to:

$$\omega_{k,t+1} = \omega_{k,t} - \alpha_\omega \cdot \omega_{k,t} \quad (2)$$

while mean and variance are unchanged. Each pixel can only match a single Gaussian. If the pixel falls under more than one Gaussian a priority parameter, named *Fitness*, selects the matched Gaussian. The Fitness (*F*) parameter is given by:

$$F_k = \omega_k / \sigma_k \quad (3)$$

When a Gaussian has an high *F* value it represents a background pixel with high probability. The *K* Gaussians are sorted in decreasing order of the *F* value (that is, $F_1 > F_2 > F_3 \ldots$). The Gaussian that matches with the pixel and has the highest *F* is considered as the matched distribution. A specific "no match" updating procedure is executed when the pixel does not match any Gaussian. In this case the parameters of the Gaussian with smallest *F* are updated as:

$$\begin{aligned} \mu_{k,t+1} &= pixel \qquad \sigma^2_{k,t+1} = variance\_init \\ matchsum_k &= 1 \quad \omega_{k,t+1} = 1 \Big/ \sum_k matchsum_k \end{aligned} \quad (4)$$

where *variance_init* is a fixed initialization value. The weights of the other Gaussians are decremented as in (2) while their means and variances are unchanged.

B. Background identification

The background identification is performed using the following algorithm, where *T* is a prefixed threshold:

$$B = \arg\min_b \left(\sum_{k=1}^{b} \omega_{k,t} > T \right) \quad (5)$$

.Eq. (5) works adding in succession the weights of the first *b* F sorted Gaussians, until their sum is greater than *T*. The Gaussians that verify (5) represent the background. If a pixel matches with one of these Gaussians, it is considered as background pixel, otherwise it is classified as foreground.

III. Hardware Implementation

The GMM computational complexity is very high and grows with the number of Gaussians used for each pixel. Also the accuracy of the background identification increases with this number. It has been observed that a good compromise is a model with 3 Gaussians.

The proposed circuit implements the single channel, luminance based, OpenCV implementation of the GMM algorithm.

The computational complexity of the algorithm has a growth rate of O(n) where n is the frame size. The target of this work is to process HD video at more than 10 fps. It is then required to process 21Mpps. A software implementation does not reach these performances. The proposed HW implementation of the GMM has been described in VHDL code. The target devices for the implementation have been high performance Virtex FPGA. Our GMM processor complies with the OpenCV algorithm allowing a simple replacement of a slower software implementation in a real time image processing system.

A. Circuit

The proposed circuit is shown in Fig. 1. Input data are the 8 bit luminance of the input pixel (Pixel), and the statistical model of the pixel (Parameters). The output is, for the input pixel the updated statistical model (Updated Parameters) and the background/foreground tag. A detailed explanation of the algorithm implemented by each circuital block of Fig. 1 is given in the following. **Fitness**: computes the Fitness factor (3) for the three Gaussians; **Match**: verifies the match

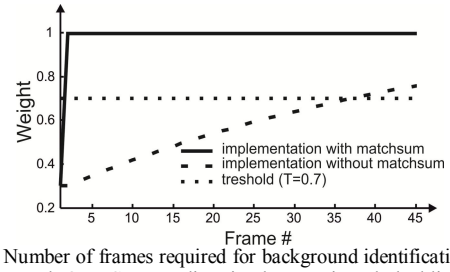

Fig. 2. Number of frames required for background identification. Solid line: proposed, OpenCV compliant implementation; dashed line: GMM algorithm of [13]

condition for the three Gaussians; **Control logic**: sorts the Gaussians in decreasing Fitness order and establishes which Gaussian is updated as in (1), (2), (4). Gaussian sorting is performed through a two by two comparison of the *F* factors. Only three comparators and few logic gates are needed. **Parameter Update**: if the match condition is verified, *Weights*, *Mean* and *Variance* blocks update the parameters as described in (1), (2). If no Gaussian matches the pixel, *No-Match* block updates mean, variance and weight of the smallest F Gaussian using (4). **Output Selection**: Depending on whether the match condition is verified or not, the logic of output selection establishes the values of the updated parameters. **Background identification**: verifies the background identification condition shown in (4) and generates the bg/fg mask for each frame. The OpenCV algorithm uses, for each Gaussian, a counter named *matchsum* that is incremented when the related distribution matches with a pixel and is used to update the weight when no match is verified. In [13], the matchsum is not used and, when no match is verified, the weight is initialized to a fixed (very small) value. OpenCV GMM method is faster when the initial identification of the background is conducted. It detects the background after one frame while the GMM algorithm of [13] requires about 35 frames as shown in Fig 2.

The introduction of the $matchsum_{k,t}$ entails the synthesis of three counters that are not on the critical path. The only drawback is an increase of circuit area.

B. Bandwidth

The crucial point of the GMM is the memory size required to record the background parameters. For each pixel mean, variance, weight and matchsum of each distribution must be loaded/stored. Depending on their wordlengths, memory bandwidth bottlenecks can incur.

In the software OpenCV implementation the parameters are double precision (64 bits) floating point numbers. The parameter bits for each pixel are therefore 758. If, as example, the memory throughput is 128 bit per cycle, 6 clock cycles are needed in order to load the parameter bits for each *pixel*. The consequence, also due to the hardware complexity, is that real time video processing is not possible using floating point arithmetic.

A detailed examination of the GMM algorithm reveals that most signals have a limited dynamics. Mean, weight and variance range in [0,255], [0,127], and [0,1], respectively. In this case using a fixed point representation provides good performances while reducing HW complexity and the required bandwidth towards the memory.

Table I Representations used in our implementation

Parameter	Representation	Bit #
α_ω= 0.02	$U_{-6,9}$	4
$\mu_{k,t}$	$U_{7,4}$	12
$\sigma_{k,t}$	$U_{13,-2}$	12
$\omega_{k,t}$	$U_{-1,8}$	8

The number of bit of the fixed point representation of the parameters is based on both their range and the required accuracy. If the mean of the Gaussians is a 23 bit number (7 bit are for the integer part, 18 for the fractional part) both memory bandwidth and logic utilization are very high. In order to improve the performances the wordlengths have been reduced obtaining good precision with the least possible number of bits. It is worth noting that the number of bits of the fractional part of the parameters are also a function of the learning rate ($\alpha_{G,t}$, α_ω). Smaller learning rates increase the bits needed for the fractional part. As example if α_ω=0.02, at least 6 bits are needed for the fractional part of α_ω. Using lower wordlengths, causes an underflow that results in α_ω=0 and impairs the GMM algorithm. The representations used in our implementation are shown in Table I, given with Um,n notation. Um,n indicates an unsigned fixed point number where 2^m is the weight of the msb and 2^{-n} is the weight of the lsb. With regard to the matchsum, it has been observed that 4 bits per matchsum allow good performances with reduced logic utilization.

With the proposed representations 116 bits are employed for each pixel, greatly reducing the required memory bandwidth for real time image processing. To process 10 fps with frame size of 1920X1080 the memory bandwidth is 287 MB/s. Moreover, the images in Fig. 3 show that the double precision floating point implementation (Fig. 3a) is very similar to the proposed implementation (Fig. 3b).

C. Nonlinear functions

Different nonlinear functions are required for the implementation of the GMM. As example, the fitness factor, the weight in (4), and the learning rate α_G require the binary inversion operation. The actual choice, in order to minimize circuit complexity and maximize circuit speed, has been the use of ROM implementation of the non linear functions. The ROM implementation is only feasible if the number of input bits (and hence of the different ROM entries) is limited, since

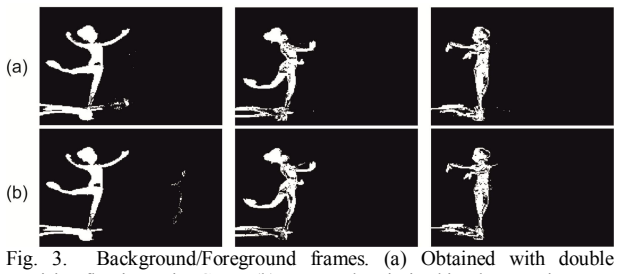

Fig. 3. Background/Foreground frames. (a) Obtained with double precision floating point GMM (b) Proposed optimized implementation.

LUT occupation exponentially increases with the number of input bits. Main advantage of the proposed HW implementation is that the input data have an optimized fixed point representation that reduces the number of input bits while providing a reliable and efficient circuit.

As example, the calculation of the Fitness factor, has been implemented with a ROM and a multiplier. The ROM stores the inverse of the standard deviation. If the variance and the inverse of the standard deviation are represented on 12 and 8 bits, respectively, the size of the ROM for the Fitness computation is 2^{12}X8 bits. The logic utilization for Virtex5 xc5vlx50 is 243 Slices out of 28800. Working frequency and power are 191.6 MHz and 1.87 mW, respectively.

D. Reconfigurability

As previously noted, the number of bits of the Gaussians parameters must be chosen according to the α_ω value that is a function of the considered background (slow background changing scenes require lower α_ω values).

The proposed design is highly parameterized and allows a straightforward modification of the representation of the parameters. This allows a fast adaption of the circuit to new applications and is of utmost importance for commercial applications since provides optimal designs with the lowest possible number of bits.

E. Results and Performaces

The proposed, not pipelined, implementation of the OpenCV algorithm has been synthesized and implemented on various Xilinx FPGA. The only sequential elements are the FFs that synchronize input and output data. The circuit has been tested using artificial video, computer animated videos with simple background and using video sequences taken from real surveillance cameras. The circuit performs optimally and runs smoothly without showing reliability problems. Table II shows the performances of different circuit implementations varying the α_ω value and the number of bits of the representations. As shown, all the proposed implementations satisfy the 10fps requirement. For decreasing α_ω values the hardware complexity increases while speed decreases. However for α_ω lower than 0.002 hardware complexity, as well as the speed of the circuit, do not increase in a sensible way. The use of a cheaper Virtex-4 FPGA reduces the speed of the circuit by an amount equal to 27% . As a consequence it might be useful to exploit pipelining to increase performance in order to use even cheaper FPGAs.

IV. Conclusions

This paper presents an hardware implementation of the GMM algorithm used in the OpenCV. The proposed circuit implemented on Virtex5 (xc5vlx50 Speed grade -3) allows a maximum working frequency of 47 MHz and is hence able to process 22 fps for an HD video with frame size 1920x1080. The circuit has reduce logic utilization of 1572 slice lut (5.5% of the available LUT). The dynamic power dissipation is 27.6 mW. Having implemented the OpenCV algorithm the circuit allows a very fast initialization of the background.

The algorithm has been implemented using a fixed point arithmetic instead of the double precision floating point representation used in the OpenCV library. In this paper it is shown that the precision of the resulting system is unchanged.

References

[1] D. Gutchess, M. Trajković, E. Cohen-Solal, D. Lyons and A. K. Jain,"A background model initialization algorithm for video surveillance", in *Proc. Eighth IEEE International Conference on Computer Vision*, vol. 1, Vancouver, BC, July 2001, pp. 733-740.

[2] I.Haritaoglu, D. Harwood and L.S. Davis, "A fast background scene modeling and maintenance for outdoor surveillance", in *Proc. of the 15th International Conference on Pattern Recognition*, vol. 4, Barcellona, Spain, 2000, pp. 179-183.

[3] B. Gloyer, H. K. Aghajan, K. Y. Siu, and T. Kailath,"Video-based freeway monitoring system using recursive vehicle tracking," in *Proc. SPIE Symp. Electronic Imaging: Image and Video Processing,* 1995, pp. 173-180.

[4] L. Vibha et al, "Moving Vehicle Identification using Background Registration Technique …", Proc. of the Intern. MultiConference of Engineers and ComputerScientists, vol. 1, Hong Kong, 2008, pp. 572-577.

[5] Z. Chaohui, D. Xiaohui, X. Shuoyu, S. Zheng and L. Min, "An Improved Moving Object Detection Algorithm Based on Frame Difference and Edge Detection", in *Proc. Fourth International Conference on Image and Graphics*, Chengdu, 2007, pp. 519–523.

[6] R. Mingwu and S. Han, "A Practical Method for Moving Target Detection Under Complex Background", *Computer Engineering*, 2005, pp. 33-34.

[7] Z. Yunchu, L. Zize, L. En and T. Min, "A Background Reconstruction Algorithm Based on C-means Clustering for Video Surveillance", *Computer Engineering and Application*, 2006, pp. 45-47.

[8] S. Y. Chien, S.Y. Ma and L.G. Chen," Efficient moving object segmentation algorithm using background registration technique", *IEEE Trans. Circuits Syst. Video Technol*., 2002, pp. 577--586.

[8] C. Ridder, O. Munkelt and H. Kirchner, "Adaptive background estimation and foreground detection using Kalman filtering", In *Proc. ICAM*, 1995, pp. 193-199.

[10] K.P. Karmann, A. Brandt, "Moving object recognition using an adaptive background memory", *In Proceedings of Time-Varying Image Processing and Moving Object Recognition*, vol. 2, V.Capellini, Ed., 1990.

[11] J. Toyama, J. Krumm, B. Brumitt and B. Meyers, "Wallflower: Principles and practice of background maintenance", in International Conf. on Computer Vision, Kerkyra, Greece , 1999, pp. 255–261.

[12] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-Time Tracking of the Human Body", *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997, vol 19, no 7, pp. 780-785.

[13] C. Stauffer, W. Grimson, "Adaptive background mixture models for realtime tracking", *Proc. IEEE conf. Computer Vision and Pattern Recognition*, 1999,

[14] H. Jiang, H. Ardö, V. Öwall, "Hardware accelerator design for video segmentation … ", in Proc. ISCAS, 2005, vol. 2, pp.1142-1145.

[15] OpenCV library on source forge: "http://sourceforge.net/projects/opencvlibrary/"

Table II Performances varying the α_ω value and the FPGA.

αω	FPGA	Fmax (MHz)	Fps @1024X1024	Fps @1920X1080	Dynamic Power (mW)	Slice LUT
0.02	Virtex 5 xc5vlx50	47	44	22	27.6	1572
0.002	Virtex 5 xc5vlx50	40	38	19	27.6	2098
0.02	Virtex 4 xc4vfx12	34	32	16	51.3	2601