Improved Real-time Implementation of Adaptive Gassian Mixture Model-based Object Detection Algorithm for Fixed-point DSP Processors

Byung-eun LEE, Thanh-binh NGUYEN, Sun-tae CHUNG
(*School of Electronics Engineering, University of Soongsil, Seoul* 156-743, *Korea*)

Abstract – **Foreground moving object detection is an important process in various computer vision applications such as intelligent visual surveillance, HCI, object-based video compression, etc. One of the most successful moving object detection algorithms is based on Adaptive Gaussian Mixture Model (AGMM). Although AGMM-based object detection shows very good performance with respect to object detection accuracy, AGMM is very complex model requiring lots of floating-point arithmetic so that it should pay for expensive computational cost. Thus, direct implementation of the AGMM-based object detection for embedded DSPs without floating-point arithmetic HW support cannot satisfy the real-time processing requirement. This paper presents a novel real-time implementation of adaptive Gaussian mixture model-based moving object detection algorithm for fixed-point DSPs. In the proposed implementation, in addition to changes of data types into fixed-point ones, magnification of the Gaussian distribution technique is introduced so that the integer and fixed-point arithmetic can be easily and consistently utilized instead of real number and floating-point arithmetic in processing of AGMM algorithm. Experimental results shows that the proposed implementation have a high potential in real-time applications.**

Key words – ***background modeling***; ***real-time computing***; ***object detection***

Manuscript Number: 1674-8042(2010)02-0116-05

dio: 10.3969/j.issn.1674-8042.2010.02.04

1 Introduction

Foreground moving object detection is an important process in various computer vision applications such as intelligent visual surveillance, HCI, object-based video compression, etc.[1,8]. Intelligent visual surveillance requires analyzing interesting objects' activities. Interesting objects are foreground moving objects different from background objects in the background scene. It is now well known that the performance of intelligent visual surveillance heavily depends on how precisely and rapidly interesting objects are detected[1]. A popular approach to moving object detection is the one based on background subtraction where background is modeled and foreground pixels are determined by matching pixels of an incoming video frame against background model. Thus, accurate background modeling is required for successful performance of the object detection. However, scenes may be continuously changing due to dynamic background objects (ex. swaying tree' branches, escalators, fountains), noises, variations of illumination, and etc. so that more accurate but more complex scene modeling like adaptive statistical scene modeling have been proposed[3,5-7]. Although these scene modeling have produced much better performance with respect to object detection rate, they need heavy computational burden. Since object detection stage has been investigated as the heaviest computing part among whole stages in intelligent visual surveillance, real-time efficient implementation of these object detection algorithms adopting statistical scene modeling are important, especially for embedded systems which have limitation in computational resources like CPU power and memory size.

DSPs are popularly adopted for implementing image processing algorithms due to its superior ability in digital signal processing, but many popularly adopted DSPs do not support H/W floating-point ALU but floating-point emulation S/W library in fixed-point ALU.

In this paper, we propose a novel real-time implementation of Adaptive Gaussian Mixture Model (AGMM)-based moving object detection algorithm for fixed-point DSPs like TI's DM642. Firstly, we implement an improved version of adaptive Gaussian mixture model[2] but adopting just Y (luminance; gray-level intensity) element instead of RGB color space. This approach saves computation time a lot but it still satisfies sufficient preciseness within the permitted level. However it was found

※ **Received:** 2010-03-20
Project supported: This work was supported by Soongsil University Research Fund and BK21 of Korea
Corresponding author: Thanh-binh NGUYEN(binh.nguyen@sectic.com)

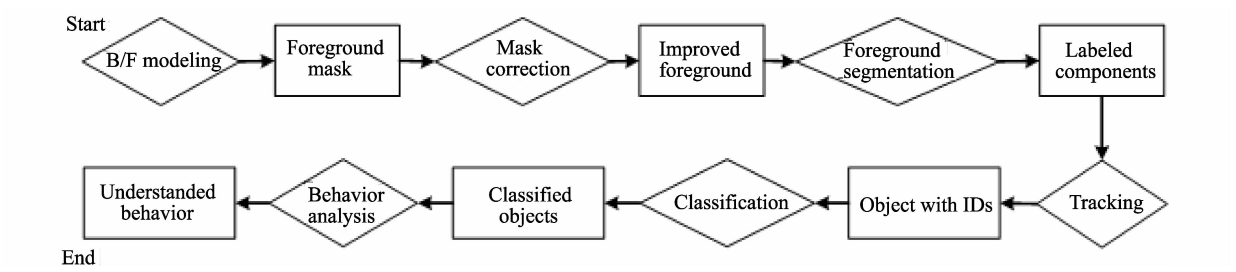

Fig. 1 General work flow of visual surveillance system based on background subtraction

not sufficient to achieve required real-time processing for DSPs without H/W floating-point ALU. For the required real-time performance in DSPs with fixed-point ALU only, we firstly introduce a magnifying Gaussian distribution technique utilizing the fact that when we magnify the parameters of a Gaussian, all of characteristics of it are still reserved in a new expanded space. Then, the fixed-point integer representation can be easily utilized instead of floating-point real number representation in implementation of AGMM. Although the range of integer numbers is not wide and smooth as floating real numbers, the tradeoff between speed and performance can be accepted in this case. The experiment results show the proposed implementation has a high potential in real-time applications.

The rest of the paper is organized as follows. Section 2 introduces processing stages including Object detection of visual surveillance, and Adaptive Gaussian Mixture Modeling, which are necessary for understanding the works of this paper. Section 3 describes our proposed approach. There, we first show the common method to transfer a computation from float to fixed point. After that, we explain Gaussian scaling solution to zoom in Gaussian shape to make it adapt to integer number. Experiment results are discussed in section 4, and finally the conclusion is presented in section 5.

2 Moving object detection and Adaptive Gaussian Mixture Model

2.1 Object detection and scene modeling

Most of intelligent visual surveillance processing based on background subtraction usually goes through the following stages[1]: scene modeling, foreground mask extraction, foreground mask correction, blob segmentation through connected component labeling, tracking, classification, behavior analysis as shown in Fig. 1.

In these stages, foreground masks means the set of all foreground pixels, and are represented as binary images where white (value 1) pixels are foreground pixels and black (value 0) pixels are background pixels. Foreground pixels are extracted from the incoming current image frame through matching it with the scene model[2]. The successful workings of the whole processing rely on the correct segmentation of moving objects which depends on the scene model. Moreover, some research showed that scene modeling is one of the most demanding computation stage in this whole process[1].

Fig. 2 shows example images related with scene modeling and foreground extraction steps.

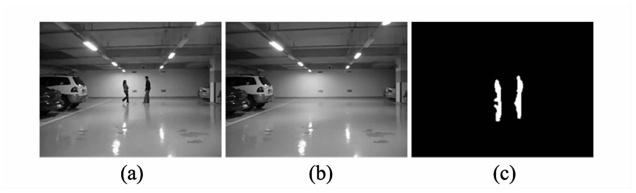

Fig. 2 Images related to scene modeling and foreground extraction steps: (a) Original frame image, (b) Background image, (c) Foreground binary image

2.2 Adaptive Gaussian Mixture Model for scene modeling

The AGMM can adapt to slow illumination changes, repetitive motions from clutter background, etc. In Ref. [3] Stauffer and Grimson model the RGB value histories of each pixels in the scene by a mixture of K Gaussian distributions, while, in our paper, we use the K Gaussian distributions to model for just the Y element in YUV format. Our implementation relies on the improvement idea for GMM of Z. Zivkovic[2]. Each pixel in the scene is modeled by a mixture of Gaussian distributions. The probability that a certain pixel has a gray-level value of x_N at time N is as

$$p(x_N) = \sum_{j=1}^{K} w_j^N \eta(x_N; \mu_j^N, \sigma_j^N), \tag{1}$$

where w_j^N is the weighting parameter of the j^{th} Gaussian component at time N. $\eta(x_N; \mu_j^N, \sigma_j^N)$ represents the Gaussian distribution with a mean μ_j^N and a variance $(\sigma_j^N)^2$ of j^{th} component at time N, that's

$$\eta(x_N; \mu_j^N, \sigma_j^N) = \frac{1}{\sqrt{2\pi}\sigma_j^N} \cdot \exp\left(-\frac{1}{2(\sigma_j^N)^2}(x_N - \mu_j^N)^2\right). \tag{2}$$

The K distributions are ordered based on the fitness value w/σ and the first B distributions are used as a model of the background of the scene where B is estimated like Eq. (3).

$$B = \arg\min_b \left(\sum_{j=1}^{b} w_j > T\right). \tag{3}$$

The threshold T is the minimum fraction of the background model. Now, background subtraction is performed by marking a foreground pixel any pixel that is more than 2.5 standard deviations away from any of the B distributions.

Parameter estimating of the Gaussian mixture model is done through online EM algorithms using expected sufficient statistics update equations in the beginning and then using L-recent window version when the first L samples are processed. The expected sufficient statistics update equations provide a good estimate at the beginning before all L samples can be collected. This initial estimate improves the accuracy of the estimate and also the performance of the tracker allowing fast convergence on a stable background model. The L-recent window update equations gives priority over recent data therefore the tracker can adapt to changes in the environment. The parameter estimation rule in the beginning by expected sufficient statistics is given in Eq.(4).

$$\begin{aligned}
\hat{\omega}_k^{N+1} &= \hat{\omega}_k^{N} + \frac{1}{N+1}(\hat{P}(\omega_k \mid x_{N+1}) - \hat{\omega}_k^{N}),\\
\hat{\mu}_k^{N+1} &= \hat{\mu}_k^{N} + \frac{\hat{P}(\omega_k \mid x_{N+1})}{\sum_{i=1}^{N+1}\hat{P}(\omega_k \mid x_i)}(x_{N+1} - \hat{\mu}_k^{N}),\\
\hat{\sigma}_k^{2^{N+1}} &= \hat{\sigma}_k^{2^{N}} + \frac{\hat{P}(\omega_k \mid x_{N+1})}{\sum_{i=1}^{N+1}\hat{P}(\omega_k \mid x_i)}\cdot\\
&\quad ((x_{N+1} - \hat{\mu}_k^{N})^2 - \hat{\sigma}_k^{2^{N}}).
\end{aligned} \tag{4}$$

Where

$$\hat{P}(\omega_k \mid x_{N+1}) = \begin{cases} 1, & \omega_k \text{ matches Gaussian component;}\\ 0, & \text{otherwise.}\end{cases}$$

While the parameter estimation rules by L-recent window version is given in Eq.(5). And in context of L-recent window, in Eq.(5) the Gaussian component that matches with current pixel is updated.

$$\begin{aligned}
\hat{\omega}_k^{N+1} &= \hat{\omega}_k^{N} + \frac{1}{L}(\hat{P}(\omega_k \mid x_{N+1}) - \hat{\omega}_k^{N}),\\
\hat{\mu}_k^{N+1} &= \hat{\mu}_k^{N} + \frac{1}{L}(\frac{\hat{P}(\omega_k \mid x_{N+1})X_{N+1}}{\hat{\omega}_k^{N+1}} - \hat{\mu}_k^{N}),\\
\hat{\sigma}_k^{2^{N+1}} &= \hat{\sigma}_k^{2^{N}} = \frac{1}{L}(\frac{\hat{P}(\omega_k \mid x_{N+1})}{\hat{\omega}_k^{N+1}}\cdot\\
&\quad (x_{N+1} - \hat{\mu}_k^{N})^2 - \hat{\sigma}_k^{2^{N}}).
\end{aligned}$$

3 The proposed real-time implementation of AGMM-based object detection algorithm

Although using one color channel Y (luminance) instead of 3 RGB color channels saved processing time in AGMM implementation, most DSPs without floating-point HW ALU still lack more CPU computing power to process the implementation of AGMM-based moving object detection algorithm in the required real-time. The main reason is that the AGMM algorithm heavily employs floating-point arithmetic which is usually done by very costly emulation routines in DSPs without H/W floating-point support. The usual remedy to this is to change floating-point arithmetic into fixed-point arithmetic with some sacrifice in numerical error. In this paper, in addition to changes of data types into fixed-point ones, we magnify the Gaussian distribution so that the integer numbers and fixed-point arithmetic can be easily utilized instead of real numbers and floating-point arithmetic in processing of AGMM algorithm.

3.1 Binary scaling and Q-format

Binary scaling is a computer programming technique used mainly by embedded C, DSP and assembler programmers to perform a pseudo floating point using integer arithmetic. It is both faster and more accurate than directly using floating point instructions. However care must be taken not to cause an overflow. A position for the virtual binary point is taken, and then subsequent arithmetic operations determine the resultants binary point. Binary points obey the mathematical laws of exponentiation.

To give an example, a common way to use integer math to simulate floating point is to multiply the coefficients by 65 536. This will place the binary point at B16. For instance to represent 1.2 and 5.6 floating point real numbers as B16 one multiplies them by 2^{16} giving 78 643 and 367 001. The multiplying of these together gives 28 862 059 643. To convert it back to B16, divide it by 2^{16}. This gives 440400B16, which when converted back to a floating point number (by dividing again by 2^{16}, but holding the result as floating point) gives 6.719 99. The correct floating point result is 6.72.

Another common method used to convert from floating-point number to integer representation is Q-format. Q-format is a fixed-point number representation format where the number of fractional bits (and optionally the number of integer bits) is specified. For example, a Q15 number has 15 fractional bits; a Q1.14 number has 1 integer bit and 14 fsactional bits.

Both of methods were performed by shifting a floating number to a new suitable range in integer space. They got the similar problems about overflow and bias during processing. One needs to care the number of shift bit or the range that they choose to present floating point. We will extract this idea to improve GMM by presenting it in integer space.

3.2 Magnifying Gaussian model

The AGMM algorithm is based on a mixture of Gaussian distributions. Characteristics of Gaussian distribution are determined by its mean and variance (standard deviation). Thus, when value x is scaled, then scaling the mean and standard deviation in the same scale will keep the characteristics of Gaussian distribution in the same way. This fact can be seen easily as follows.

Let's scale μ and σ by R when x is scaled by R. Then, the Gaussian distribution (2) will be changed as

$$\frac{1}{\sqrt{2\pi}\sigma_j^N R}\exp\left(-\frac{1}{2(\sigma_j^N R)^2}(x_N R - \mu_j^N R)^2\right) = \frac{1}{R}\eta(x_N; \mu_j^N, \sigma_j^N).$$

From the above analysis, we determine that it is possible to extend scope of x_N, μ_j^N, and σ_j^N, by magnifying the mean and variance but still with the properties of Gaussian distribution kept.

We want to map the Gaussians from floating-point real number domain to fixed-point integer representation domain by scaling technique that have similar idea as binary scaling mentioned above. Fig. 3 illustrates to the mapping model.

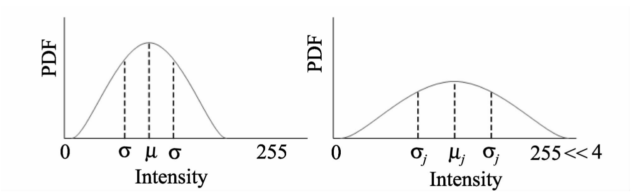

Fig. 3 Gaussian scaling model

In our implementation, the data space of intensity [0,255] will be mapped to [0,255≪4]. In this case, the distance between x_N and x_{N-1}, $d = x_N - x_{N-1}$ should belong to [0,255≪4]. Standard distance which must be calculated to compare to variance, $D = d * d$ stay in range [0,(255 * 255)≪8]. We chose 4 as basic shifting bits to scale all main parameters of Gaussian for trade off between overflow and bias. The others parameter will be calculated as

$$\begin{aligned} \sigma_i^2 &= \sigma^2 \ll 8, \\ \mu_i &= \mu \ll 4, \\ w_i &= w \ll 10. \end{aligned} \qquad (6)$$

The thresholds and extra other parameters will be scaled by a suitable shifting to make sure an unchanged model. Moreover, they must be in balance of safeness and preciseness with a very strict filter.

4 Experimental results

4.1 Comparison between the proposed implementation and the conventional floating-point implementation with respect to preciseness

We compare our fixed-point AGMM implementation and conventional floating-point with respect to blob correction ability and background image. Fig. 4 shows the result which shows the proposed implementation produces almost similar output images as the conventional implementation does.

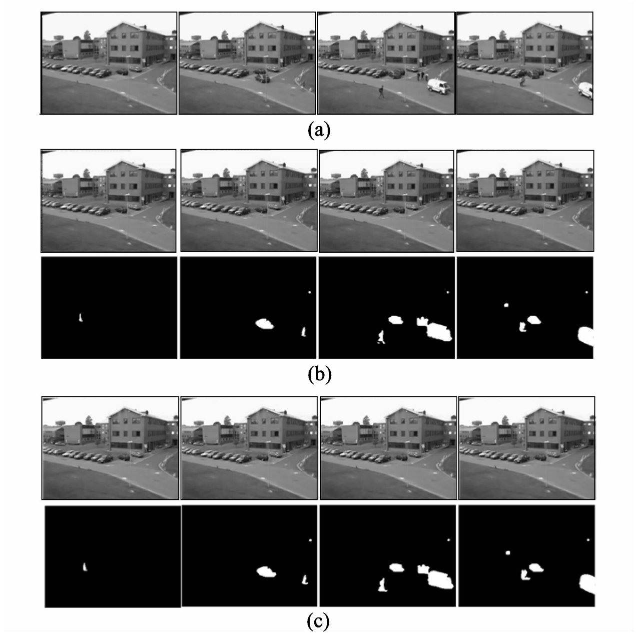

Fig. 4 (a) Frames 340, 630, 940, 1540 of Pets 2001 test video, (b) The equivelent background and foreground mask images by the proposed implemenation, (c) background and foreground mask images by the conventional implementation. Foreground mask in Fig. 4(b) and 4(c) was obtained using CLNF method that is produced in our previous works[9]. Although the value range of integer number is lower than floating domain, the result is still in permitted bias

4.2 Comparison about processing time

In order to compare the computational cost between the conventional implementation and the proposed implementation, we did two experiments in both environments, desktop computer with a 3 GHz Pentium 4 Core of 3 GHz and 3 GB main memory and DSP TI DM642 evaluation board.

First experiment calculates the processing time for background modeling in one frame, Fig. 4(a) in both the conventional and our upgrading version. And the second experiment calculates the processing time for continuous background modeling in movies. The resolution of each image frame is 320×240 and sizes of movie1 and movie2 are 4.67 MB (221 frames) and 245 MB (1 117 frames). In all testing, we just extract the time of background modeling from whole processing. Table 1 shows the results for the first and second experiment, respectively in desktop computer.

Tab. 1 Comparison of the processing time between the conventional Gaussian Mixture Model (GMM) and the proposed fixed point version for one frame of Fig. 11(a) and two movies in PC

	Frame(s)	Movie 1(s)	Movie 2(s)
Conventional GMM	0.004 226	1.247 776	5.694 340
Our GMM	0.001 482	0.504 759	2.312 254

The results of Tab. 1 certainly show that the proposed implementation is faster than the conventional implementation even they was tested in powerful floating point CPU. At the experiment in a fixed-point DSP TMS320DM642 (Frequency: 720 MHz) that is shown in Tab.2, the speed of the proposed one was found to be much faster than conventional one.

Tab.2 Comparison of the processing time between the conventional Gaussian Mixture Model (GMM) and the proposed fixed point version for one frame and two movies in DSP

	Frame(s)	Movie 1(s)	Movie 2(s)
Conventional GMM	0.009 247 7	2.052 994 5	9.630 296 9
Our GMM	0.001 132 7	0.314 259 6	1.263 025 7

5 Conclusion

In this paper, we proposed a novel real-time implementation of AGMM-based object detection algorithm for fixed-point DSPs. In addition to utilizing gray-level data as opposed to RGB color data in the conventional AGMM implementation and changes of data structures and floating-point arithmetic operations, magnification of Gaussian model through parameter scaling is introduced which enables fixed-point arithmetic operations instead of floating-point arithmetic operations so that it saves a lot of computational time. The experiment results show the effectiveness of the proposed implementation.

References

[1] T. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranov, V. Eruhimov, 2005. Computer vision workload analysis: case study of video surveillance systems. *Intel Technology Journal*, 9(2): 109-112.

[2] P. KadewTraKuPong, R. Bowden, 2009. An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection. Proc. 2nd European Workshop on Advanced Video-Based Surveillance Systems.

[3] C. Stauffer C, W.E.L Grimson, 1999. Adaptive Background Mixture Models for Real-time Tracking. IEEE CVPR, p. 244 - 252.

[4] M. Onoe, 1981. Real Time Parallel Computing Image Analysis. Plenum Press, New York.

[5] D. Koller, J. Webber, T. Huang, J. Malik, G. Ogasawara, B. Rao, S. Russel, 1994. Towards Robust Automatic Traffic Scene Analysis in Real Time. Proceedings of International Conference on Pattern Recognition.

[6] A. Elgammal, R. Duraiswami, D. Harwood, L. Davis, 2002. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. Proceeding of IEEE, p.1151-1163.

[7] K. Toyama, B. Brumitt J. Krumm, B. Meyers, 1999. Wallflower: Principles and Practices of Background, Greece. Proceeding of the 7the IEEE International Conference on Computer Vision, p.255-261.

[8] O. Javed, M. Shah, 2008. Automated Multi-Camera Surveillance: Algorithms and Practice, Springer, Heidelberg.

[9] B. Nguyen, C. Sun Tae, 2009. An Improved Real Time Blob Detection for Visual Surveillance. CISP.