Hardware Accelerator Design for Video Segmentation with Multi-modal Background Modelling

Hongtu Jiang
Digital ASIC Group, CCCD, Department of Electroscience, Lund University
Håkan Ardö
Department of Mathematics, Lund University
Viktor Öwall
Digital ASIC Group, CCCD, Department of Electroscience, Lund University

Abstract— **Among many of the algorithms for video segmentation, one based on statistical background model [1] was developed with the unique feature of robustness in multi-modal background scenarios. However, with large amount of calculations due to pixel wise processing of each frame, such an algorithm could only achieve a low frame rate far from real-time requirements on computers. In this paper, a hardware accelerator is proposed, with a dedicated architecture aiming to address the computation as well as memory bandwidth demand. The whole system is targeted to an FPGA platform, which is served as a real-time test bench where long term effects caused by fixed point quantization and various parameter settings can be studied. Meanwhile, memory bandwidth as well as memory size are investigated and reduction by up to 60 percent through similarity exploitation for neighboring Gaussian parameters is envisioned. Furthermore, an controller synthesis tool is used to relieve the effort for the manual design of complex control unit, scheduling the operations of the whole system.**

I. Introduction

Video applications are omnipresent. They are essential for industries such as surveillance, communications and entertainment. Among these applications, moving object segmentation is one of the fundamental steps to many automated video analysis tasks. For example, the video coding standard MPEG-4 [2] relies on the decomposition of each frame of an image sequence into video object planes to improve coding efficiency and support content based functionality. Other applications include automated traffic surveillance system, in which segmentation is regarded as the essential step prior to later analysis such as vehicle tracking, accident detection etc.

Among many of the early alternative segmentation algorithms, the ones based on either inter-frame difference or comparison of current frame with a reference background image were widely used. While such algorithms may work to some extent in short terms, they are error prone when dealing with varying background scenes, e.g. illumination variation over time, an entering object stopping in the scene. Besides, using inter-frame difference always results in larger detected object areas and fails to detect slowly moving object. Nowadays, with the substantial increase of the computation capacity, such non-adaptive background modelling are abandoned by most researchers. More and more complex background estimation methods, adaptive to real world situations, have been proposed.

II. Video segmentation Algorithms

Chien et al [3] developed an adaptive background model using so called background registration approach. In their paper, they assume that the longer a pixel remains stationary, the more probable that it belongs to the background. By counting whether a pixel stays approximately in the same value for a predefined period, a new background pixel is registered to the background memories where the old value is discarded. In this way, background plane is updated progressively and the moving object is detected by thresholding the difference between the current frame and the registered background plane. A number of other adaptive background models have also been reported [4]–[6]. Although these algorithms produce better modelling towards real world scenarios by background learning process, most of them fail to deal with multi-modal background distribution. A multi-modal background distribution is caused by repetitive background object motion, for example, swaying trees, reflections of the lake surface, flickering of the monitor etc. As the pixel, lying in the region where repetitive motion occurs, will generally consists of two or more background colors, the RGB value of that specific pixel changes over time. This would result in false foreground object detection by most adaptive background estimation approach mentioned above.

In [1], a background model based on multi-modal pixel distribution is proposed to address the issue. By representing each pixel process using a mixture of Gaussian distributions, repetitive background motions are merged into one of the several background distributions for each pixel. However, as the algorithm processes video stream pixel-wise by updating several Gaussian distributions for each pixel, the calculation burden in parameter updating is unbearable for computers in real time applications. In [1], only a frame rates of 11-13 frames/s is obtained even for small frame size of 160x120 on an SGI O2 workstation. For real time video applications with larger frame size, a dedicated hardware architecture seems to be a must. However, as far as the authors knowledge, no such hardware implementation has been reported before. Furthermore, issues emerge with regard to memory bandwidth and storage when it comes to implementation, which is quite common to most video/image processing task. Since the update of background distribution is slow most of the time in a slowly changing scene, the wordlength of each parameter tends to grow to fulfil the increasing dynamic range. With a reasonable frame size of 352x288 that is used throughout this paper, and under the assumption that 3 Gaussian are used for each pixel process, approximately 6 MB data have to be updated for each frame. This imposes a huge demand for calculation as well as memory bandwidth and size. In this paper, a dedicated hardware architecture is developed aiming to address all the issues mentioned. With an FPGA platform, simulations can be accomplished in real-time to observe long term effects resulting from fixed point quantization as well as parameter settings. In addition, a controller synthesis tool is developed based on [9] to reduce the design effort for controller design.

The paper is organized as follows. In Section III, the mixed Gaussian models for multi-modal background is explained. The hardware architecture is presented in Section IV. In SectionV, an introduction to the controller design flow is shown. Finally, the results are presented in section VI, and conclusions are drawn in Section VII.

0-7803-8834-8/05/$20.00 ©2005 IEEE.

III. Gaussian Mixture Model

A Gaussian Mixture model was proposed independently by Stauffer and Grimson [1] and Friedman and Russell [12] to model each pixel in the video sequences based on statistical approach. The value of each particular pixel in the image frames over time can be considered as a pixel process. If such pixel lies in the region belonging to a stationary background object, a single Gaussian distribution will suffice to model the pixel process while accounting for acquisition noise. A single Gaussian distribution with varying mean value and variance would be enough to adapt to the changes, If only lighting condition changes over time on the pixel. However, in real world situations, background pixels may consist of several background object values over time. Consider the visual field covered by swaying trees from time to time. The pixel value, originally representing the color of the leave, would change to value of the other background object when the leave moves away, for example, the road color. Thus, the value of such specific pixel over time would include two independent adaptive Gaussian processes representing both the leave color and the color of the other background object.

In order to meet the needs for modelling multi-modal pixel process, a mixture of Gaussian distributions are assigned to each one of the pixels in the image sequences. The whole procedure is formulated as follows. A pixel process, defined as an collection of most recent measurements *I*(x,y,t), where *I* is the image sequence, can be viewed as a mixture of several independent noise processes and thus modelled using the sum of Gaussian distributions with weighting factors. The probability of observing the current pixel value is

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}),$$

where K is the number of Gaussian distributions and ω is the weighting factor. The mixture of Gaussian distributions are ordered according to ω/σ in decreasing order. The portion of Gaussian distributions considered as background process is defined by

$$B = argmin_b \left(\sum_{k=1}^{b} \omega_k > H \right),$$

where H is the predefined parameter determining how much proportion of the mixture distributions accounts for the background process. When this parameter is small enough, the pixel process in fact becomes a single adaptive Gaussian distribution. To update the parameters of each Gaussian, each new pixel value is checked against the Gaussian mixture. When a match is found, the weight, mean and variance values of the matched distributions are updated respectively as follows:

$$\omega_{k,t} = (1-\alpha)\omega_{k,t} + \alpha$$

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t$$

$$\sigma^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t);$$

where α, ρ is the learning factor. A match is defined as a pixel value within 2.5 standard deviations of a distribution. For those unmatched, the weight is updated according to

$$\omega_{k,t} = (1-\alpha)\omega_{k,t},$$

while the mean and the variance remain the same. The basic idea behind such updating scheme is: when the pixel belongs to the background, it will usually match one of the distributions, and more evidence is building up upon the matched distribution by its increasing weight. The mean value is adjusted toward the current pixel value, which is in fact an adaptation process in case illumination condition changes. If the pixel represents the foreground object, it would not, in most cases, match any of the distributions. In this case, the least probable distribution would be replaced with a distribution with the current value as its mean value, an initially high variance and low prior weight. If the object stops moving, the replaced distribution is matched for a few subsequent frames, and evidence continues building up upon that distribution until it finally becomes a background process, thus a new added object is properly incorporated into the background scene. Otherwise the new distribution would keep being replaced by new pixel values till a new background is discovered.

Once the background distributions are defined, a segmented binary image containing only moving object can be accomplished by checking each pixel value against the corresponding background distributions. Such binary image forms the basics for the subsequent analysis work, eg. noise cancelling, object identification.

IV. Hardware Implementation

Maintaining a mixture of Gaussian distributions for each pixel is costly in terms of both calculation capacity and memory storage, especially with large frame size. To cope with streaming RGB data from video camera in real time, a dedicated hardware architecture is proposed to be used as a simulation platform with streamlined data flow. On the platform, effects of the number of Gaussian distributions and parameter properties can be studied. Therefore, the maximum number of distributions is set to 9, while for real applications, the number of Gaussian distributions can be lowered substantially, e.g 2 or 3 in total. For simplification, a architecture for 5 Gaussian distributions per pixel is shown in Fig. 1, which is basically the same architecture as that of 9 Gaussian distribution . At each clock cycle, an incoming RGB pixel is read into the matching logic from the input buffer. Together with the parameters for the corresponding mixture of Gaussian distribution, a matching process starts. The Gaussian distributions are checked against the incoming pixel in parallel, resulting in a vector consisting of matching signals. To avoid the competition of several Gaussian distributions matching the same input pixel, only the one with the most evidence (highest ω/σ) is selected as the matching distribution. In the design, a switching logic is included in the matching logic, deciding on the right matching distribution and rearranging all the Gaussian distributions. After the matching block, the only matched distribution is switched to the bottom among all Gaussian distributions. Depending on whether the new incoming pixel matches any of the Gaussian distribution, the parameter updating logic updates the matched distribution accordingly. If none of the distributions are matched, the mean value of the distribution in the bottom is replaced by the RGB color of the incoming pixel. The variance value is replaced with a predefined large value while the weight is kept intact. In this way, the re-normalization of the weights are avoided, leading to simplified hardware implementation. On the other hand, if a match is spotted, multiplications and square root calculations are performed to update mean and variance values. In this paper, multiplications and square roots are pipelined into 2 and 3 stages respectively.

The Gaussian distributions after update stages have to be sorted for use in the next frame. All distributions should be ordered according to their evidence (ω/σ). In order to fully utilize hardware parallelism to enhance the sorting speed, various parallel sorting networks are investigated. Parallel sorting networks [7], [8], [10] have been a research topic among the academic society for the last three decades. Many trade-offs have been made regarding sorting speed and number

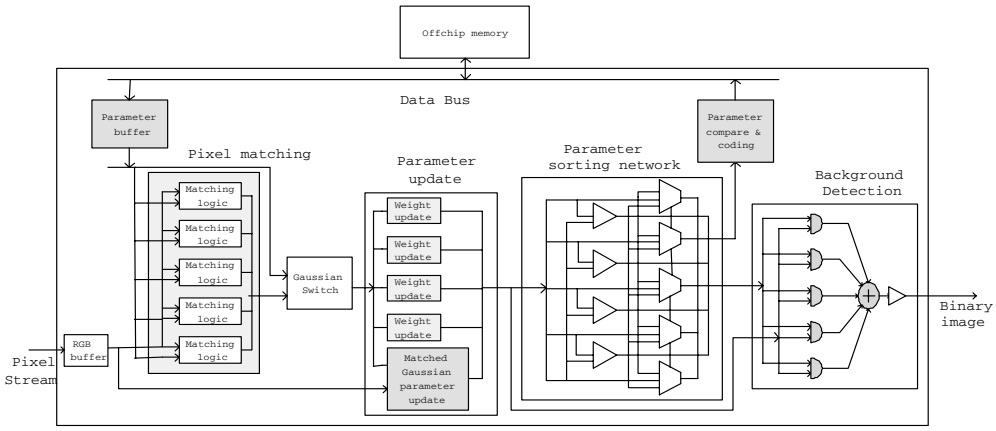

Fig. 1. Simplified Datapath Architecture with only 5 Gaussian distributions per Pixel

of sorting elements employed in the network. For example, bitonic sort is one of the fastest sorting networks [7] while in [8] only $O(nlog(N))$ comparators are used to sort an N input elements.

By the observation that only one Gaussian distribution is updated at a time, as long as all the distributions are initially in the right order, the sorting of N Gaussian distributions can be reduced into inserting an updated distribution among N-1 ordered distributions. As a result, both number of sorting stages and number of comparators are reduced substantially to only one sorting stage with N-1 comparators and N MUXes. The benefits from both a speed and an area considerations are obtained by removing the redundancy in other existing sorting networks. The architecture for the sorting is shown in Fig. 1.

Whenever all the Gaussian distributions are sorted in the right order, background detection could start. By adding up the weights of all the Gaussian distributions that have more evidence than the updated one, a binary pixel value is generated depending on whether the sum is greater than a predefined parameter H. Like the input pixel value, a binary segmented pixel is streamed out at one bit each clock cycle.

For slow background updating process, large dynamic range for each parameter in the distribution is required to record slight changes. Storing parameters with big wordlength for all Gaussian distributions incurs heavy memory bandwidth bottlenecks. Consider the background model with minimum Gaussian distributions, which contains three Gaussian distributions with two for bi-model background and one for foreground. C++ simulation shows that both mean and variance varies for the same Gaussian distribution between different frames are in the order of 10^{-4}. For fixed number representation, 28 bits are assigned to each RGB mean parameters, among which 8 bits accounts for integer part while 20 bits for fractional part. The same number of bits goes for the fractional part of the variance but only 4 bits are needed for integer part. Together with 16 bits for the weight parameter, one Gaussian distribution contributes to 124 bits. In real time environment with 25 frames/s and a frame size of 352x288, the memory bandwidth for reading and writing Gaussian parameters goes up to nearly 225 MB/s. Without data compressing scheme, such high demand for memory bandwidth would make it difficult to implement on some FPGA platforms. In this paper, a data compression scheme is proposed which utilizes similarities for Gaussian distributions in adjacent areas. Inspired by the fact that adjacent pixel values are very close to each other and many video segmentation algorithm works on pixel blocks instead of on each pixel, there could exist a way to classify "similar" Gaussian distributions in adjacent area. In practice, each Gaussian distribution can be regarded as a three dimensional cube in the RGB space, where the center of the cube is composed of RGB mean values whereas the border to the center is specified by 2.5 times variance value. One way to measure the similarity between two distributions is to check how much volume the two cubes overlaps, which can be performed by inspecting the deviation of two cube centers in regard to the border length. The reason for such criteria lies in the fact that a pixel that belongs to one distribution will most likely belong to the other if they have much overlapping volume. Whenever the deviation value in any dimension is greater than 2.5 times the sum of two variances that is defined by the deviation factor, there is no overlapping volume. In fact, a factor K, can be used as follows to determine the degree to which two cubes are adjacent to each other.

$$\mu_{1R} - \mu_{2R} \leq 2.5K(\sigma_1 + \sigma_2)$$

$$\mu_{1G} - \mu_{2G} \leq 2.5K(\sigma_1 + \sigma_2)$$

$$\mu_{1B} - \mu_{2B} \leq 2.5K(\sigma_1 + \sigma_2), K \subset [0, 1]$$

Two distributions falls within the criteria are regarded as the same. By saving only "different" distributions with the count of the "same" succeeding distributions, memory bandwidth are reduced. From simulation in C++, different K value are chosen to evaluate the reductions for memory bandwidth. Obviously, with larger K value specified, more savings could be accomplished. However, more noise would be generated in the binary image due to increasing error matches in the matching phase . Fortunately, such noise is non-accumulating and can be erased by morphology process [11] with appropriate specified K value. In this paper, a K value of 0.35 is specified, with memory bandwidth savings of up to 60% reported from C++ simulation. In order to further decrease the wordlength, variable word length coding algorithm such as Huffman coding is under consideration in future implementations.

V. Flexible design with Controller synthesis

There is yet another advantage of implementing the design in FPGA. Due to its data as well as computation intensive characteristics, simulating the algorithm for the effects on segmentation quality with different parameter settings as well as data quantization becomes difficult to achieve on computers, in particular, for long term effects. In C++, the simulation time for one minute video takes about twenty minutes. Evaluating long terms effects on computers, for example, the

Fig. 2. Original and Segmented sample images

effects of different numbers of Gaussian distributions in the mixture in one day period, would take weeks to accomplish. The situation gets even worse if fixed point quantization noise are considered. With FPGA implementation, all simulation are done in real-time.

To fully coordinate and schedule all kinds of operations in a flexible design, a dedicated a controller unit is required. It is crucial for reusing the same hardware architecture to evaluate different parameters. In order to reduce the effort of the manual design of a controller, an controller synthesis tool is developed based on [9]. The tool takes in the behavioral description of the datapath architecture defining the available set of micro-operations, and a microprogram written in C-like input syntax that contains the algorithm with additional declarations such as memories. As an output a complete controller with module descriptions and interconnection specifications is generated. since architecture extensions to basic FSM could result in optimized controllers in specific application, a range of controller architectures are supported and can be accessed through architecture option specification before the synthesis starts. For the current application, a controller architecture with incremental circuitry is considered [9]. In this architecture, the branch address calculation within the same block of code, composed of only sequential statements, is performed by the hardware incrementer. At the end of a block a non-incremental branch address is calculated by the control logic and a select branch signal is set. This architecture is particularly suitable for algorithms that have long stretches of sequential statements, i.e. the next state generation logic in the basic FSM architecture can be replaced by an incrementer and mux which is more power efficient. With the tool, more complicated architectures can be implemented freely based on user requirements and algorithm structures since it is a fully automated synthesis process.

VI. Results

A dedicated hardware architecture is proposed for the video segmentation algorithm capable of handle multi-modal background conditions. The whole system is targeted to Xilinx Virtex2 1000 FPGA platform and has been implemented in VHDL. From the synthesis results, the system is ready to run on 40MHz due to its fully streamlined architecture. This will result in 38 FPS for video sequences with big frame size of 1024×1024. C++ simulations indicate that memory bandwidth savings of up to 60% could be achieved. Sample images from the original video sequence and segmented binary video sequence is shown in Fig. 2. Together with peripheral frame grabber and SDRAM controller, the future FPGA platform is capable of evaluating the long term effects caused by various factors, such as number of Gaussian distributions per pixel, wordlength of each Gaussian parameters, etc.

VII. Conclusions

A hardware architecture for a video segmentation algorithm based on statistical background modelling has been presented. The architecture provides a calculation capacity allowing real-time processing of relatively large images, 1024×1024, at a frame rate of 38 FPS. To reduce the large memory bandwidth required for storing Gaussian parameters, a specific memory scheme is proposed. Substantial memory bandwidth reductions are envisioned by utilizing similarities for adjacent Gaussian distributions. On the other hand, to cope with the task for processing large amount of data, which makes a software simulation approach impossible for studying long term effects, a testbench based on an FPGA platform has been developed, capable of real time evaluation of the system performance as well as parameter properties. With controller synthesis tool, the effort for manual design of the control unit is reduced substantially.

Acknowledgment

This work is partially sponsored by *Axis Communications* (www.axis.com), and all simulations are performed with the image data obtained through their network camera AXIS 2120.

References

[1] C. Stauffer, W. Grimson, "Adaptive background mixture models for real-time tracking", Proc. IEEE conf. Computer Vision and Pattern Recognition, 1999.
[2] *www.mpeg4.net*, 2004.
[3] S. Chien, S. Ma, L.Chen, "Efficient Moving Object Segmentation Algorithm Using Background Registration Technique", *IEEE tran. on circuits and systems for video technology*, July, 2002.
[4] Y. Tsaig, A. Averbuch, "Automatic Segmentation of Moving objects in video sequences: A region labeling approach", *IEEE tran. circuits and systems for video technology*, July, 2002.
[5] N. Li, et al, "Real-time video object segmentation using HSV space", Proc. International conf. Image Processing, 2002.
[6] D. Gao, Z. Zhou, "Adaptive background estimation for real-time traffic monitoring", Proc. IEEE conf. Intelligent Transportation Systems, 2001.
[7] K.E. Batcher, "Sorting Networks and their Applications". Proc. AFIPS Spring Joint Comput. Conf., 1968.
[8] M. Ajtai, J. Komlos, S. Szemeredi, "An O(N log N) Sorting Network". Proceedings of the 25th ACM Symposium on Theory of Computing, 1983.
[9] H. Jiang, V. Öwall, "FPGA Implementation of Controller-Datapath Pair in Custom Image Processor Design", Proc. ISCAS, 2004
[10] G.V.Russo, M.Russo, "A novel class of sorting networks", *IEEE tran. circuits and systems I: Fundamental Theory and Applications*, July, 1996.
[11] H. Hedberg, F. Kristensen, P. Nilsson, V. Öwall, "A Low Complexity Architecture For Binary Image Erosion And Dilatation Using Structuring Element Decomposition", Proc. ISCAS, 2005
[12] N. Friedman, S. Russell, "Image segmentation in video sequence", Proc. 13th conf. Uncertainty in A.I., 1997.
[13] S. Chien et al, "Single Chip Video Segmentation System with Programmable PE array", Proc. IEEE Asia-Pacific Conference on Asic, 2002.