A Hardware Architecture for Real-Time Video Segmentation Utilizing Memory Reduction Techniques

Hongtu Jiang, Håkan Ardö, and Viktor Öwall, *Member, IEEE*

***Abstract*—This paper presents the implementation of a video segmentation unit used for embedded automated video surveillance systems. Various aspects of the underlying segmentation algorithm are explored and modifications are made with potential improvements of segmentation results and hardware efficiency. In addition, to achieve real-time performance with high resolution video streams, a dedicated hardware architecture with streamlined dataflow and memory access reduction schemes are developed. The whole system is implemented on a Xilinx field-programmable gate array platform, capable of real-time segmentation with VGA resolution at 25 frames per second. Substantial memory bandwidth reduction of more than 70% is achieved by utilizing pixel locality as well as wordlength reduction. The hardware platform is intended as a real-time testbench, especially for observations of long term effects with different parameter settings.**

***Index Terms*—Field-programmable gate array (FPGA), mixture of Gaussian (MoG), video segmentation.**

I. Introduction

AUTOMATED video surveillance systems have been gaining substantial interests in the research community in recent years. This is partially due to the progress in technology scaling that enables more robust yet computationally intensive algorithms to be realized with reasonable performance. The advantage of surveillance automation over traditional closed circuit TV (CCTV)-based system lies in the fact that it is a self contained system capable of automatic information extraction, e.g., detection of moving objects and tracking. The result is a fully or semi automated surveillance system, with the potential of increased usage of mounted cameras and the reduced cost of human resources for observing the output. Typical applications may include both civilian and military scenarios, e.g., traffic control, security surveillance in banks or antiterrorism.

Crucial to all these applications is the quality of the video segmentation, which is a process of extracting objects of interest (foreground) from an irrelevant background scene. The foreground information, usually composed of moving objects, is passed on to later analysis units, where objects are tracked and their activities are analyzed. A wide range of segmentation algorithms have been proposed in the literatures, with robustness aimed for different situations. In [1], comparisons on segmentation qualities are made to evaluate a variety of approaches. Table I shows five segmentation algorithms that are cited by many literatures, namely frame difference (FD) [2]–[5], median filter [6]–[8], linear predictive filter (LPF) [1], [9]–[12], mixture of Gaussian (MoG) [13]–[19] and kernel density estimation (KDE) [20]. Using FD, background/foreground detection is achieved by simply observing the difference of the pixels between two adjacent frames. By setting a threshold value, a pixel is identified as foreground if the difference is higher than the threshold value, otherwise background. The simplicity of the algorithm comes at the cost of the segmentation quality. In general, bigger regions than the actual moving part are detected as foreground area. Also it fails to detect inner pixels of a large, uniformly-colored moving object, a problem known as aperture effect [3]. As more sophisticated algorithms are utilized aiming for improved robustness and segmentation quality, the complexity of realizing such systems increases.

In fact, no perfect system exists to handle all kinds of issues within different background models. For realistic implementation of such systems, trade-offs have to be made between system robustness (quality) and system performance (frame rate, resolution, etc.). A background model based on pixel wise multimodal Gaussian distribution was proposed in [13] with robustness to multimodal background situations, which are quite common in both indoor and outdoor environments. From Table I it can be seen that the KDE approach has the highest segmentation quality which however comes at the cost of a high hardware complexity and even to a larger extent, increased memory requirements. These facts, has led us to choose the MoG approach for the developed system. A multimodal background is caused by repetitive background object motion, e.g., swaying trees, flickering of the monitor, etc. As a pixel lying in the region where repetitive motion occurs, its value will consist of two or more background colors, i.e., the RGB value of that specific pixel changes over time. This will result in false foreground detection in most other approaches. Various modifications to the algorithm for potential improvements are reported [14]–[19]. However, none of these works address the issue of algorithm

Manuscript received March 26, 2007; revised February 08, 2008. First published December 09, 2008; current version published January 30, 2009. This work was supported in part by VINNOVA Competence Center for Circuit Design (CCCD). This paper was recommended by Associate Editor C. N. Taylor.

H. Jiang was with the Department of Electrical and Information Technology, Lund University, SE-22100 Lund, Sweden. He is now with the Ericsson Mobile Platforms, SE-22370 Lund, Sweden (e-mail: Hongtu.Jiang@ericsson.com).

H. Ardö is with the Centre for Mathematical Sciences Lund University, SE-22100 Lund, Sweden (e-mail: ardo@maths.lth.se).

V. Öwall is with the Department of Electrical and Information Technology, Lund University, SE-22100 Lund, Sweden (e-mail: viktor.owall@eit.lth.se)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2008.2009244

1051-8215/$25.00 © 2009 IEEE

TABLE I
COMPARISON OF DIFFERENT SEGMENTATION ALGORITHMS

	FD	Median	LPF	MoG	KDE
Algorithm performance	Fast	Fast	Medium	Medium	Slow
Memory requirement	1 Frame	$50-300$ Frames	1 Frame of mean	1 Frame of k Gaussian parameters	n Frames of k Gaussian parameters
Segmentation quality	Worst	Low	Acceptable	Good	Best
Hardware complexity	Very low	Medium	Low to medium	Low	High

performance in terms of meeting real-time requirements with reasonable resolution. In [13], only a frame rate of 11–13 fps is obtained even for a small frame size of 160 × 120 on an SGI O2 workstation. In our software implementation on an AMD 4400+ processor, a frame rate of 4–6 fps is observed for video sequences with 352 × 288 resolution. In addition to performance issues, we have found no studies on possible algorithm modifications that could lead to potentially better hardware efficiency.

In this paper, we present a dedicated hardware architecture capable of real-time segmentation with VGA resolution at 25 fps. Preliminary results of the architecture were presented in [21]. The architecture is implemented on Xilinx VirtexII pro Vp30 FPGA platform together with a variety of memory access reduction schemes, which results in more than 70% memory bandwidth reduction. Furthermore, various modifications to the algorithm are made, with potential improvements of hardware efficiency.

The choice of an FPGA as the target platform is mainly motivated from the possibility to perform algorithm changes in late stages of the system development, provided by the reconfigurability aspect. Furthermore, FPGAs gives us real time performance, hard to achieve with DSP processors, while limiting the extensive design work required for application specific integrated circuits (ASICs). A rather early overview (1999) of computer vision algorithms on FPGAs can be found in [22] while a more recent evaluation can be found in [23]. Even with the difference of 6 years, one common conclusion is that dedicated architectures are in many cases needed to achieve required performance and that reconfigurable platforms are a good way to achieve it at reasonable design time and cost. Even though advanced EDA flows exist, there is a considerable amount of required knowledge regarding hardware architecture design, often not available in image processing groups. A cooperation between theoretical and hardware research is therefore a healthy mix. In this work an automated surveillance system has been chosen as the target applications. We have found no hardware architectures for the chosen algorithm that can be used as direct comparison. However, other applications in vision systems include robotic control [24], [25], medical imaging [26], and stereo processing [27]. One common challenge is memory size and bandwidth, as in the presented design often solved with external memories.

The paper is organized as follows. Sections II and III discuss the original algorithm and possible modifications for hardware efficiency. The hardware architecture is presented in Section IV, together with the memory bandwidth reduction scheme explained in detail. Finally, the results and conclusions are covered in Sections V and VI.

II. GAUSSIAN MIXTURE BACKGROUND MODEL

In this section the used algorithm is briefly described, for a more thorough description we refer to [13]. The algorithm is formulated as follows: Measured from consecutive video frames, the values of a particular pixel over time can be regarded as a stochastic process. At any time t, what is observed for a particular pixel at x_0, y_0 is a collection of the most recent measurements over time

$$\{X_{t_0}, \dots, X_t\} = \{I(x_0, y_0, i) : t_0 \le i \le t\} \tag{1}$$

where I is the image sequence. To model such a process, a Gaussian distribution can be used. Characterized by its mean and variance values, the distribution represents a location centered at its mean values in the RGB color space, where the pixel value is most likely to be observed over frames. A pixel containing several background object colors, e.g., the leaves of a swaying tree and a road, can be modeled with a mixture of Gaussian distributions. Each distribution i, has a weight, $\omega_{i,t}$, that indicates the probability of matching a new incoming pixel, x_t. The probability of observing the current pixel value is

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} \cdot \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{2}$$

where K is the number of Gaussian distributions and η is a Gaussian probability density function. Furthermore, $\omega_{i,t}$ is the weighting factor, $\mu_{i,t}$ is the mean value and $\Sigma_{i,t}$ is the covariance matrix of the ith Gaussian at time t, which takes the form of

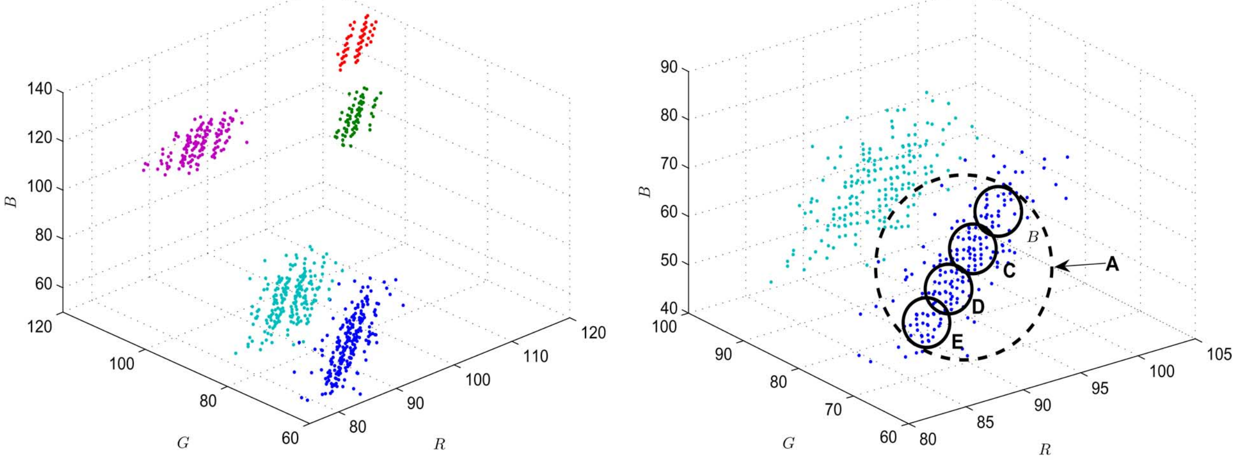
$$\Sigma_{i,t} = \sigma_i^2 \cdot \mathbf{I}. \tag{3}$$

K is determined by available resources concerning hardware complexity and memory resources.

A match is defined as the incoming pixel within J times the standard deviation off the center. In [13] J is set to 2.5, a value that has also been used in our application. The higher the weight, the more likely the distribution belongs to the background. The portion of the Gaussian distributions belonging to the background is defined to be

$$B = \min_b \left\{ b \mid \sum_{k=1}^{b} \omega_k > H \right\} \tag{4}$$

where min is used to calculate the minimum number of Gaussian distributions, b, that satisfies the condition in which the sum of the weights, ω, is less than predefined parameter H, i.e., measuring the minimum portion of the distributions that should be



(a) Five background pixel distributions in the RGB color space.

(b) A closer look at the two distributions on the bottom in (a).

Fig. 1. Background pixel distributions in the RGB color space. Instead of Sphere like distributions, each pixel cluster is rather cylindrical.

accounted for by the background. If a small value is chosen for H, the background model is usually unimodal.

If a match is found, the matched distribution is updated as

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha \quad (5)$$

$$\mu_{k,t} = (1-\rho)\mu_{k,t-1} + \rho X_t \quad (6)$$

$$\sigma^2_{k,t} = (1-\rho)\sigma^2_{k,t-1} + \rho(X_t - \mu_{k,t})^T(X_t - \mu_{k,t}); \quad (7)$$

where μ, σ are the mean and standard deviation respectively while α is the learning factors and $\rho = \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \cdot \alpha$. The mean, variance and weight factors are updated frame by frame. For those unmatched, the weight is updated according to

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} \quad (8)$$

while the mean and the variance remain the same. If none of the distributions are matched, the one with the lowest weight is replaced by a distribution with the incoming pixel value as its mean, a low weight and a large variance.

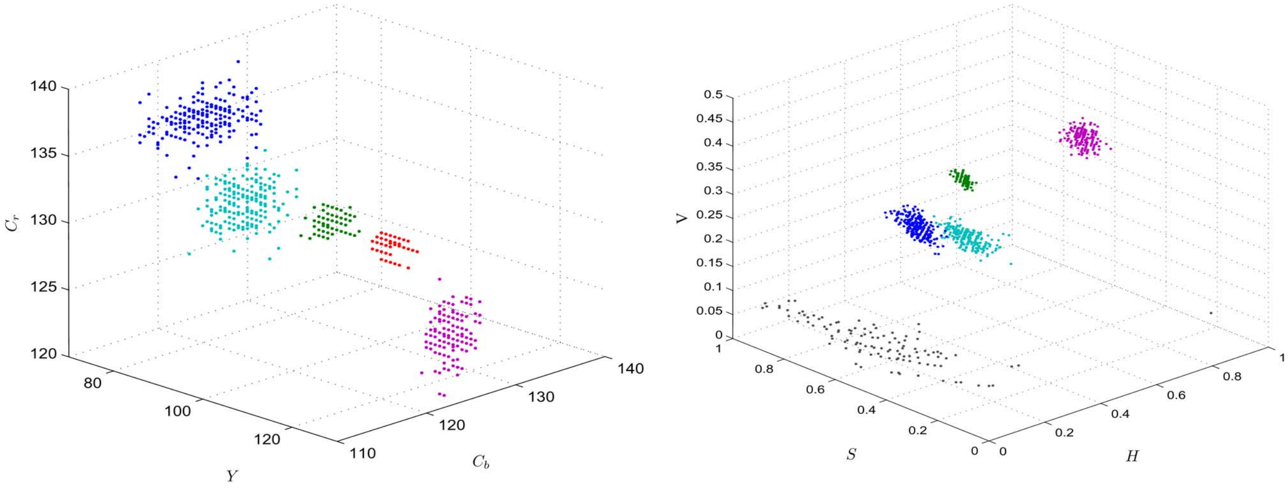
Algorithm Modifications

The algorithm works efficiently only in controlled environments. Issues regarding algorithm weaknesses in different situations are addressed in many publications [14]–[16], [18]. In this section, instead of mainly focusing on improving robustness, we propose several modifications to the algorithm with the major concern on their impact on potentially improved hardware efficiency.

A. Color Space Transformation

In theory, multimodal situations only occur when repetitive background objects are present in the scene. However, this is not always true in practice. Consider an indoor environment where the illumination comes from a fluorescence lamp. An example of a video sequence from such an environment is taken given recorded in our lab, where five pixels are picked up evenly from the scene and measured over time. Their RGB value distributions are drawn in Fig. 1(a). From the figure it can be seen that instead of five sphere like pixel distributions, the shapes of the pixel clusters are rather cylindrical. Pixel values tend to jump around more in one direction than another in the presence of illumination variations caused by the fluorescence lamp and camera jitter. This should be distinguished from the situation where one sphere distribution is moving slowly towards one direction due to slight daylight changes. Such a case is handled by updating the corresponding mean values in the original background model. Without an upper bound for the variance, the sphere describing the distribution tends to grow until it covers nearly every pixel in the most distributed direction, thus taking up a large space such that most of it does not belong to the distribution [sphere A in Fig. 1(b)]. A simple solution to this problem is to set an upper limit on the variance, e.g., the maximum value of the variance in the least distributed direction. The result is multimodal distributions represented as a series of smaller spheres (spheres B-E in the same figure). Although a background pixel distribution is modeled more precisely by such a method, additional Gaussian distributions are inferred which are hardware costly in terms of extra parameter update and storage. In [28] D. Magee proposed a cylindrical model to address the issue with primary axes of all distribution cylinders pointing at the origin. However, more parameters are needed for each cylindrical distribution than for the spherical counterpart, i.e., the parameters of a cylindrical distribution contains distance, two angles, a diameter and the height. Furthermore, it is a hardware costly computation to transform RGB values to cylindrical coordinates, e.g., division and square root. In addition, not every distribution cylinder is oriented towards the origin, see the left middle distribution in Fig. 1(a).

To be able to model background pixels using a single distribution without extensive hardware overhead, color space transformation has been investigated. Both HSV and YC_bC_r spaces are investigated and their corresponding distributions are shown in Fig. 2. By transforming RGB into YC_bC_r space, the correlation among different color coordinates are mostly removed, resulting in nearly independent color components. With varying illumination environment, the Y component (intensity) varies



(a) Five sphere distributions in the YC_bC_r color space.

(b) Unpredictable distributions in the HSV color space.

Fig. 2. Five distributions in YC_bC_r and HSV color spaces. (a). Most pixel distributions are transformed from cylindrical pixel distributions in RGB color space into sphere like pixel distributions in YC_bC_r color space. This is due to the correlation that exists among different color components in RGB color space is almost removed in YC_bC_r color space. (b). With correlated color space as well, HSV color space is no better than RGB color space, unpredictable pixel distributions appears occasionally.

the most accordingly, leaving C_b and C_r components (chromaticity) more or less independent. In [29], this property is utilized for shadow reduction. Consequently, values of three independent components of a pixel in YC_bC_r color space tends to spread equally. As shown in Fig. 2(a), most pixel distributions are transformed from cylinders back to spheres, capable of being modeled with a single spherical distribution. The transformation from RGB to YC_bC_r is linear, and can be calculated according to the following:

$$Y = 16 + 65.481 \times R + 128.553 \times G + 24.966 \times B \quad (9)$$

$$C_b = 128 - 37.797 \times R - 74.203 \times G + 112.0 \times B \quad (10)$$

$$C_r = 128 + 112.0 \times R - 93.786 \times G - 18.214 \times B. \quad (11)$$

Only minor hardware overhead with a few extra multipliers and adders are introduced, where multiplication with constants can be further utilized to reduce hardware complexity. Simplifications can be performed to further reduce the number of multiplications to 4 [30]. The HSV color space, on the other hand, also with correlated coordinates, is no better than RGB color space if not worse. Unpredictable pixel clusters appeared occasionally as shown in Fig. 2(b), which is impossible to model using Gaussian distributions.

Color space transformation has also been performed on outdoor video sequences where similar results have been observed. These results are also in line with [28].

Algorithm Simplifications

We propose two simplifications to the algorithm. In the original algorithm specification, unbounded growing distribution will absorb more pixels. As a result, the weight of that distribution will soon dominate all others. To overcome this, in [13], all updated Gaussian distributions are sorted according to the ratio ω/σ. In this way, the distribution with dominant weight but large variance does not get to the top, identified as background distribution. In our approach, with YC_bC_r color space transformation, no upper bound is needed. All distributions can be simply sorted by their weights only, effectively eliminating division operations in the implementation.

Another simplification made in the process of foreground/background detection is that instead of using (4), the determination can be made by checking the weight of each distribution separately. This is due to the fact that one pixel cluster will not spread out in several distributions by the color space transformation to YC_bC_r. The set of distributions belonging to the background is modified to be

$$\{k \,|\omega_k > H\}. \quad (12)$$

III. Hardware Architecture

To perform the algorithm with VGA resolution in real-time, a dedicated hardware architecture, with a streamlined data flow and memory bandwidth reduction schemes, is implemented to address the computation capacity and memory bandwidth bottlenecks. Due to memory bandwidth limitations, only three Gaussians are utilized which has been shown to be sufficient in our test sequences and aught to be sufficient for many environments, i.e., two background and one foreground distribution. The architecture in itself does not put any restriction on the number of distribution which could be increased if the environment so requires. Algorithm modifications covered in previous sections are implemented with potential benefits on hardware efficiency and segmentation quality. This is a large improvement to the previous work [31], [32], where only 352×288 resolution is achieved without any memory reduction schemes and algorithm modifications. In this section, a thorough description of the architecture of the segmentation unit is given, followed by detailed discussions of the memory reduction schemes.

To give a better understanding of the architecture, a simplified conceptual block diagram of the whole system is given in Fig. 3 to illustrate the dataflow within the system. The system

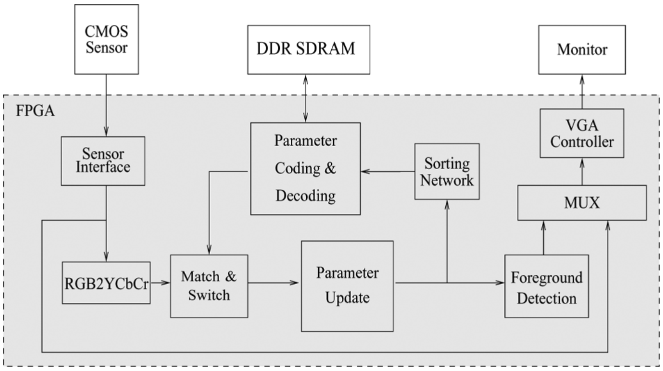


Fig. 3. Conceptual block diagram of the segmentation unit.

starts at the CMOS image sensor capturing video sequence in real-time and feeding it to the system through a sensor interface. The interface is designed to be able to control the parameters of the image sensor at run-time, e.g., analog gain and integration time, so that better image quality can be obtained within different environments. The sensor interface is also responsible for sampling the image data transferred from off-chip. In our implementation, an over sampling scheme by a higher clock frequency (100Mhz) is used to ensure the accuracy of the image data. The image data is captured with one color component at a time, and the three color components are sent to the system after serial-parallel transformation. To handle different clock frequencies, input FIFOs, implemented as distributed RAMs, are used to interface to both the segmentation logic and the VGA controller where the original video data can be monitored on a screen. The RGB values are converted into YC_bC_r components before entering the segmentation logic block, according to the algorithm modification above. Each pixel has a number of corresponding Gaussian distributions and the parameters of those are stored in off-chip memories (DDR SDRAM) due the overall amount of data. For each pixel, the Gaussian parameters are read from the DDR SDRAM and decoded by the parameter encoder/decoder. The match and switch unit checks if the incoming pixel matches any of the existing distributions. The output from this unit is reordered Gaussian distributions with the matching distribution switched to a specific port. The match and switch block is mainly composed of comparators and multiplexer. The Gaussian distributions are updated according to the algorithm modifications presented in the previous section. From this point foreground/background detection can start by checking the weight of the updated matched Gaussian distribution. The output is a binary stream to be multiplexed to the monitor, indicating foreground and background pixels with white and black colors. The updated Gaussian parameters have to be sorted for use in the next frame, and all distributions should be ordered according to their weight. This is implemented in a dedicated sorting network that will be covered in more detail in Section V. To reduce the heavy memory bandwidth incurred by accessing off-chip DDR SDRAM that stores one frame of Gaussian distributions, an encoding/decoding block is designed by utilizing pixel localities in succeeding neighboring pixels. This is covered in more detail in Section IV-C.

In the following, implementation details of the architecture shown in Fig. 4 are explained with an emphasis on the parts with algorithm modifications, indicated by shaded area. With the image data captured and transformed, the match and switch block tries to match the incoming pixel with Gaussian distributions obtained from the previous frame. To avoid the competition between several Gaussian distributions matching the incoming pixel, only the one with highest likelihood (large weight) is selected as the matching distribution. A matched Gaussian is switched to the bottom (3 in the figure). In case no matching distribution is found, a No_match signal is asserted. If there is a match, a parameter update should be performed. For the matched Gaussian distribution, a proposed updating scheme is implemented with only incrementers/decrementers for the mean and variance values. Depending on whether the incoming YC_bC_r value is larger than the mean values, a addition or subtraction is applied for parameter updating. Similar updating schemes are utilized for variance update. The proposed parameter update results in low hardware complexity by replacing the hardware costly operations in (6) and (7), e.g., square and multiplication with large wordlength with incrementer/decrementer. Other benefits of the proposed updating schemes are described in detail in Section IV-B. For the case that no match is found, a MUX is used together with the No_match signal to update all parameters for the distribution (3 in the figure) with predefined values.

A. Sorting

The updated Gaussian parameters have to be sorted for use in the next frame. In order to reduce hardware complexity found in parallel sorting networks, such as [33]–[35], while still maintaining the speed, a specific feature in the algorithm is explored. By observing that only one Gaussian distribution is updated at a time and all the distributions are initially sorted, the sorting of N Gaussian distributions can be changed to rearranging an updated distribution among $N - 1$ ordered distributions. As a result, both the number of sorting stages and the number of comparators are reduced to only one sorting stage with $N - 1$ comparators and N MUXes, resulting in both increased speed and reduced area. The architecture for a sorting network for five Gaussians is shown in Fig. 5, five distributions are used instead of three to get a more generalized architecture. From the figure, all unmatched ordered Gaussian distributions are compared with the updated distribution, i.e., three in the figure. The output of each comparator signifies which distribution is to be multiplexed to the output, e.g., if the weight of any unmatched distribution is smaller than the updated one, all unmatched distributions below the current one is switched to the output at the next lower MUX. This architecture scales very easily to support sorting more distributions since the number of stages will not increase accordingly. Since only three Gaussians are utilized in our implementation this is a trivial task. However, if future implementations require more Gaussians per pixel the sorting architecture will be useful to reduce hardware complexity. A comparison of hardware complexity between proposed sorting architecture and other schemes mentioned above is shown in

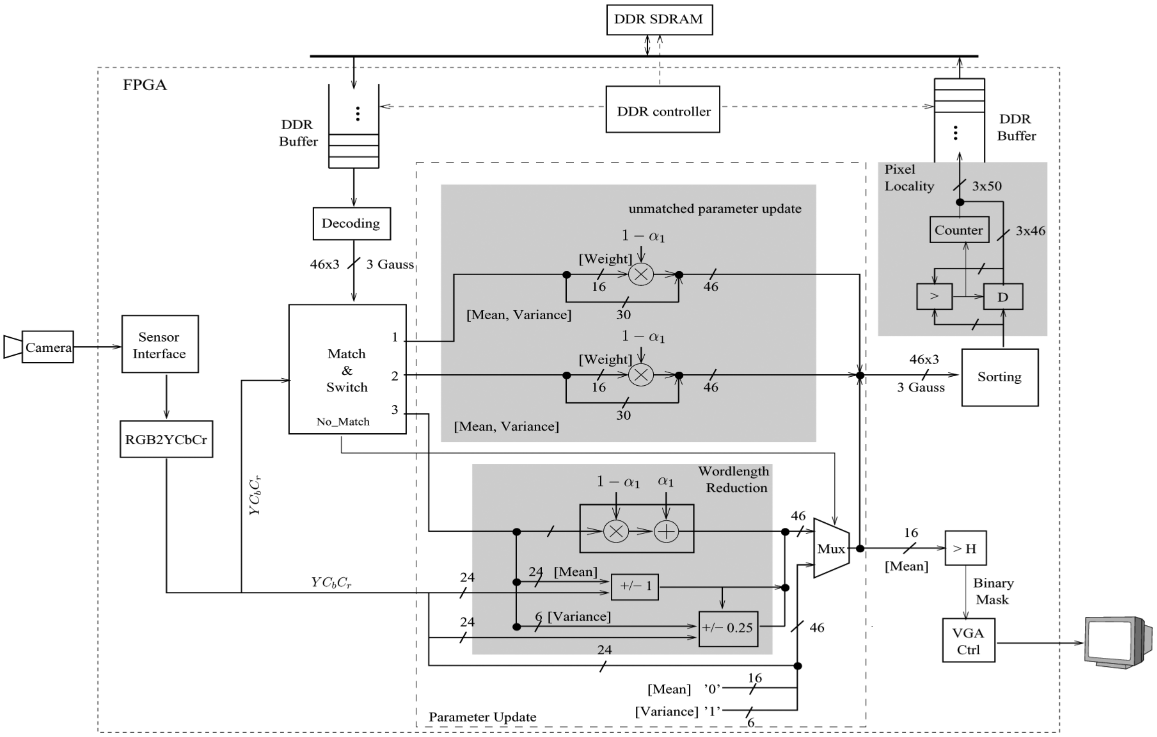


Fig. 4. More detailed architecture of the segmentation unit.

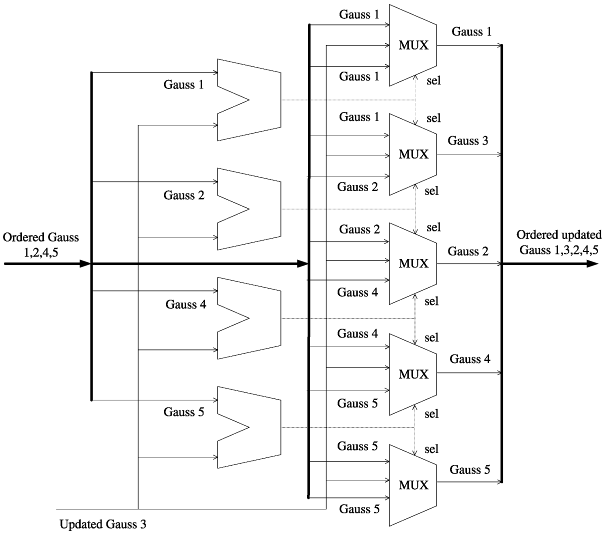


Fig. 5. Sorting architecture. Five Gaussian are sorted according to their weight. The architecture scales well with increasing number of Gaussian due to its fixed logic level (one comparator and one MUX).

Table II. The foreground detection is achieved by simply comparing the weight of the distribution on the bottom with a predefined parameter H according to the simplifications made in previous sections. All sorted Gaussian parameters are encoded and decoded before and after writing to and reading from a DDR SDRAM, with manipulated data flow controlled by a custom made DDR controller. The encoding and decoding scheme is explained in details in Section IV-C.

TABLE II
HARDWARE COMPLEXITY WITHIN DIFFERENT SORTING ARCHITECTURE, WHERE p IS $\log_2 N$

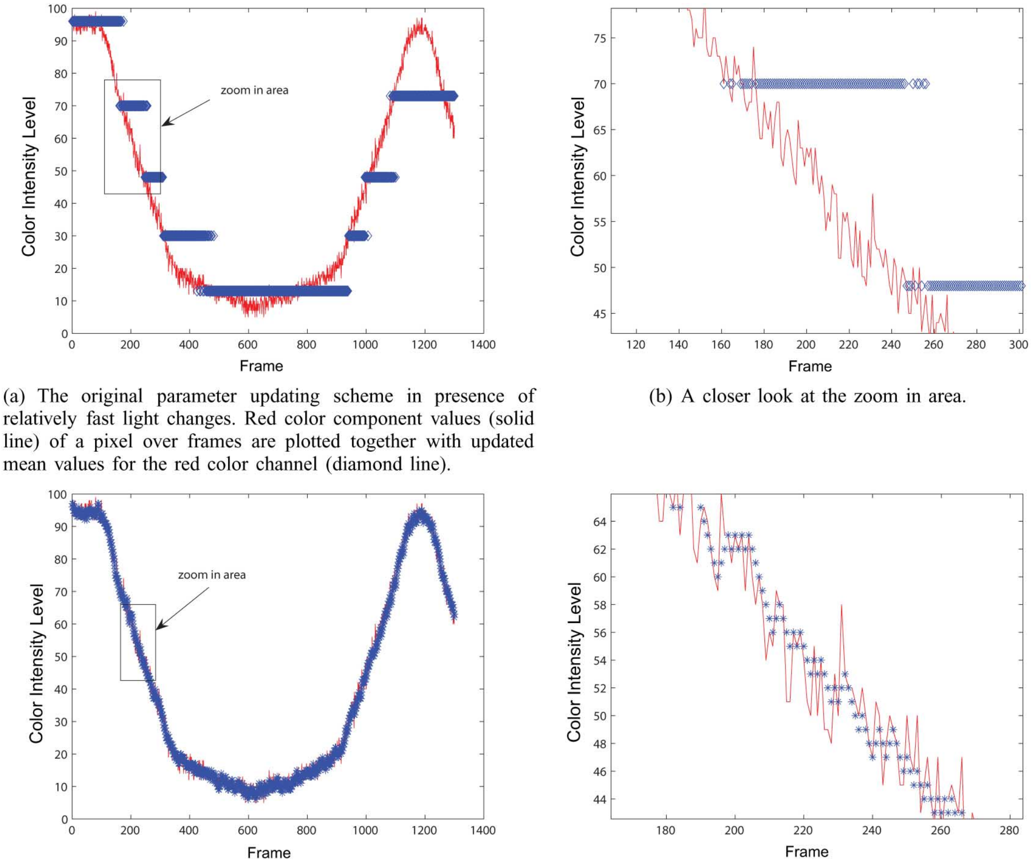
Scheme	Nr. of Comparators	Nr. of Stages
Proposed	$N-1$	2
[33]	$(p^2-p+4)2^{p-2}-1$	$(\frac{1}{4})(N)(\log_2 N)^2$
Odd-even trans. sort	$O(N)$	$O(N^2)$
Bitonic Sort	$O(N\log N)^2$	$O(\log N)^2$

B. Wordlength Reduction

Slow background updating requires large dynamic range for each parameter in the distributions, since parameter values are changed slightly between frames but could accumulate over time. According to (6) and (7), the mean and variance of a Gaussian distribution is updated using a learning factor ρ. The difference of mean and variance between current and previous frame is derived from the equation as

$$\begin{aligned}\Delta_\mu &= \mu_t - \mu_{t-1} = \rho(X_t - \mu_{t-1}) \text{ and} \qquad (13)\\ \Delta_{\sigma^2} &= \sigma_t^2 - \sigma_{t-1}^2\\ &= \rho\left((X_t-\mu_t)^T(X_t-\mu_t) - \sigma_{t-1}^2\right). \qquad (14)\end{aligned}$$

Given a small value of ρ, e.g., 0.0001, a unit difference between the incoming pixel and the current mean value results in a value of 0.0001 for Δ_μ. To be able to record this slight change, 22 bits have to be used for the mean value, where 14 bits accounts for the fractional part. Empirical results have shown that the Gaussian distributions usually are a sphere with a diameter of less than 10. Therefore, an upper bound for the variance is set to 16 and a maximum value of Δ_μ becomes $\rho \times J \times \sigma =$



(a) The original parameter updating scheme in presence of relatively fast light changes. Red color component values (solid line) of a pixel over frames are plotted together with updated mean values for the red color channel (diamond line).

(b) A closer look at the zoom in area.

(c) The proposed parameter updating scheme in presence of relatively fast light changes. Red color component values (solid line) of a pixel over frames are plotted together with updated mean values for the red color channel (star line).

(d) A closer look at the zoom in area.

Fig. 6. Results from both the original parameter updating scheme and the proposed parameter updating schemes.

$0.0001 \times 2.5 \times \sqrt{16} = 0.001$, which can be represented by 10 bits (J is the same as in Section II). Using a wordlength lower than that, no changes would ever be recorded. In practice, the bits for the fractional parts should be somewhere in between 10–14 bits and 7–14 for the mean and variance, respectively. Together with 16 bits weight and integer parts of the mean and the variance, 81–100 bits are needed for a single Gaussian distribution. To reduce this number, a wordlength reduction scheme is proposed. From (13), a small positive or negative number is derived depending on whether the incoming pixel is above or below the current mean. Instead of adding a small positive or negative fractional number to the current mean, a value of 1 or -1 is added. The overshooting caused by such coarse adjustment could be compensated by the update in the next frame. The result is that without illumination variation, the mean value will fluctuate with a magnitude of 1, negligible since the diameter of the Gaussian distribution is usually more than 10.

In a relatively fast varying illumination environment, e.g., 25 RGB value changes in a second, fast adaptation to new lighting conditions is also enabled by adding or subtracting ones in consecutive frames. Fig. 6(c) shows the experimental results of the coarse updating in a varying lighting room, where the light is turned up and down. The parameter updating scheme specified in the original algorithm is also drawn in Fig. 6(a) for comparison. A closer look at the two schemes is given in Fig. 6(d) and (b). From Fig. 6(a), it is clear that the original parameter updating does not work well in the presence of fast light changes, i.e., slow parameter updating (diamond line in the figure) will not keep track of the pixel value changes (solid line in the figure). Finally, the Gaussian distribution will not be able to match the incoming pixel values, in which case Gaussian distribution replacement takes place instead of parameter updating. The coarse updating scheme on the other hand relieves the problem to certain extent, where consecutive ones are added or subtracted to keep track of the relatively fast changes.

From these figures, it can be seen that the proposed coarse updating scheme works fine in both slow and relatively fast light changing situations. It keeps track of the relatively fast value changes in the dynamic scene while fluctuates around a constant value in the latter static scene. However, with the primary goal to reduce wordlength, the coarse parameter updating scheme results in limited improvements to the segmentation results. Nearly no visual difference can be observed in the segmented results from the two schemes.

With coarse updating, only integers are needed for mean specification, which effectively reduce the wordlength from 18–22 down to 8 bits. Similar approach can be applied to the variance

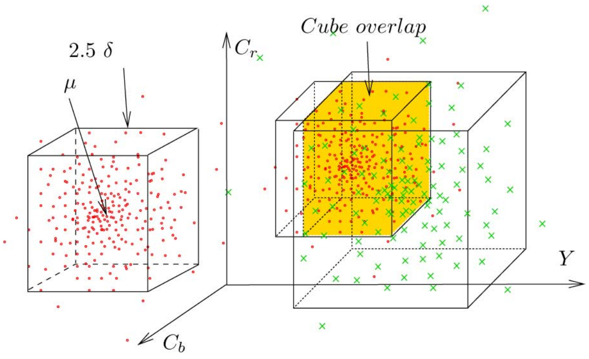


Fig. 7. Gaussian distribution similarity as modeled by cube overlapping.

(a step value of 0.25 is used instead), resulting in a wordlength of 6 bits, where 2 bits account for fractional part. Together with the weight, the wordlength of a single Gaussian distribution can be reduced from 81–100 to only 44 bits, i.e., over 43% reduction even compared to the extreme case of 81 bits. In addition, less hardware complexity is achieved as a bonus since multiplication with the learning factor ρ is no longer needed.

Thus, the proposed scheme enhance the algorithmic performance while at the same time reduce both memory bandwidth and computational complexity.

C. Pixel Locality

In addition to wordlength reduction, a data compression scheme for further bandwidth reduction is proposed by utilizing pixel locality for Gaussian distributions in adjacent areas. We classify "similar" Gaussian distributions in the following way: from the definition of a matching process, each Gaussian distribution can be simplified as a cube instead of a sphere in the YC_bC_r color space. The center of the cube is composed of YC_bC_r mean values whereas the border to the center is specified by $J = 2.5$ times the variance. One way to measure the similarity between two distributions is to check how much the two cubes overlap. If the overlap takes up a certain percentage of both Gaussian cubes, they are regarded as "similar". The whole idea is illustrated in Fig. 7. The reason for such a criteria lies in the fact that a pixel that matches one distribution will most likely match the other. The percentage is a threshold parameter that can be set to different values for different situations.

In the architecture, two similar distributions are treated as equivalent. By only saving non overlapping distributions together with the number of equivalent succeeding distributions, memory bandwidth is reduced. Various threshold values are selected to evaluate the efficiency for the memory bandwidth reduction. With a low threshold value where less overlapping Gaussian distributions are regarded as the same, more savings could be achieved. However, more noise is generated due to increasing mismatches. Fortunately, such noise is found non-accumulating and therefore can be reduced by later morphological filtering [37], [38]. Fig. 8 shows the memory bandwidth savings over frames with various threshold values. The simulation results are obtained from MATLAB, with four video sequences provided by AXIS [36] which represent both indoor and outdoor scenes. The sequences are selected to reflect a range of real-world environments with possible difficulties for many segmentation and tracking algorithms. The scene "stairs" comprises people moving up and down stairs randomly where gradual illumination changes together with shadows are the major disturbing factors. The scene "hallway" focus on a scenario with people moving closer or further away from the camera, i.e., the foreground object size is varying over time. The scene "trees" address the issue of quasi-static environments where a swaying tree is present as the dynamic background object. Finally, the scene "parklot" presents an environment with walking people and moving cars of different size. Gradual illumination as well as "waking" foreground objects is also within the focus. It can be seen from the sequences, memory reductions scheme works robustly within different real-world environments with variation only in the beginning due to varied foreground actives. During initialization phase, only background pixels are present, which exhibit high similarity within neighboring pixels. With foreground objects entering the scene, part of Gaussian distributions are replaced, which results in the decrease of number of similar Gaussian distributions. The trends will continue until it reaches a certain point where most pixel locations contain a foreground distribution. The decrease will flatten out in the end since more foreground objects always replace the distribution that represents a foreground pixel. Foreground objects activities can vary in different video scenes, e.g., continuous activities in Fig. 8(a) where people going up and down the stairs all the time, and the two peak activity periods around frames 600–900 and frames 2100–2500 in Fig. 8(b), where people walk by in two discrete time periods. In the long run, the bandwidth savings tends to stabilize (around 50%–75% depending on threshold value) after the initialization phase. Another test sequence is also experimented in our lab. Similar results are observed as shown in Fig. 9. The quality of the segmentation results before and after morphology is shown in Fig. 10, where it is clear that memory reduction comes at the cost of segmentation quality. Too low threshold value results in clustered noises that would not be filtered out by morphological filtering, which is shown in Fig. 10(c). In this implementation, a threshold value of 0.8 is selected, combined with wordlength reduction scheme, a memory bandwidth reduction of over 70% is accomplished. To evaluate long term effects of memory bandwidth reduction scheme, FPGA platform is required to collect data in real time.

IV. Results

The segmentation unit is prototyped on an Xilinx VirtexII vp30 development board, as shown in Fig. 11. The board comes with one DDR memory slot with customized on board signal traces that minimize the skew between different signals. Virtex II vp30 is equipped with two on-chip PowerPC embedded processor cores. The number of on-chip block RAMs is 136, with 2448 Kb in total. A custom-made PCB extension board is mounted on the FPGA board to support an image sensor. The whole design is partitioned into 3 clock domains and asynchronous FIFOs are used to interface blocks different clocks. A KODAK KAC-9648 CMOS sensor [39] is used to capture color images stream into the FPGA platform. Real time segmentation performance is achieved on video sequences with 3 Gaussian

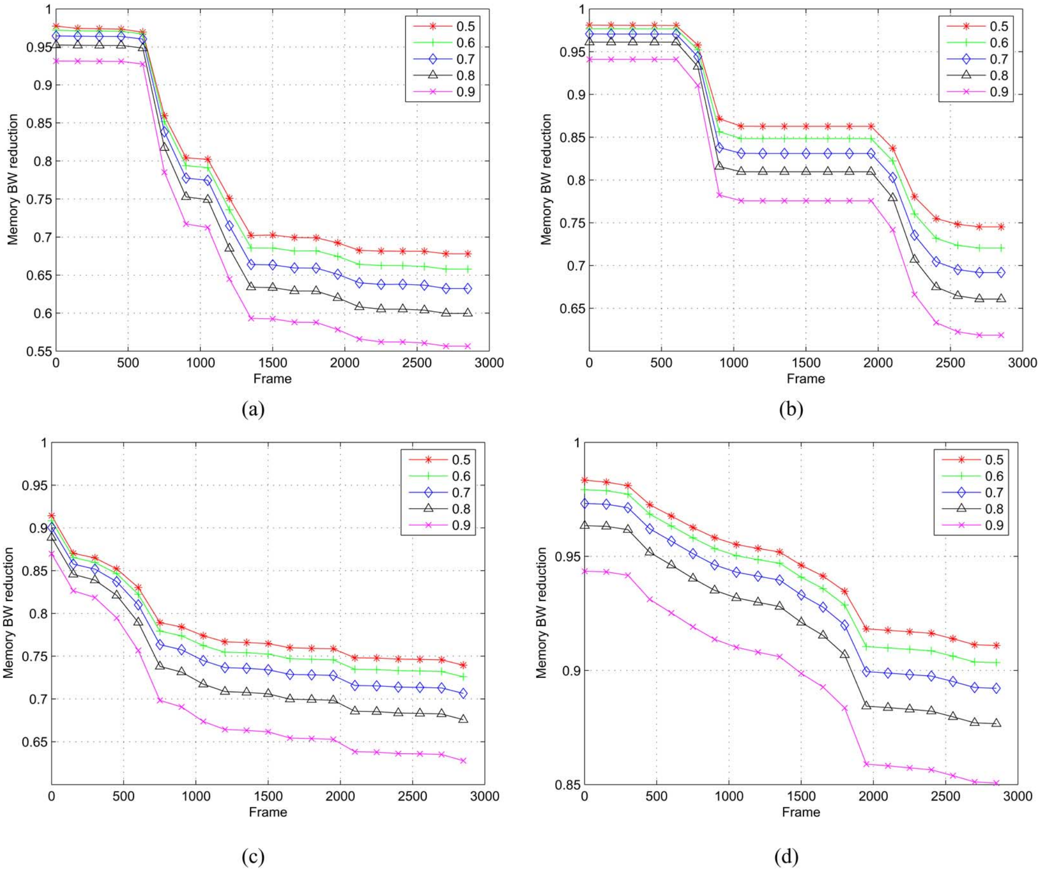


Fig. 8. Similar memory reduction results are achieved in various scenarios, both indoor and outdoor. (a) The scene "stairs" comprises people moving up and down stairs randomly. Gradual illumination changes together with shadows are the major disturbing factors. (b) The scene "hallway" focus on a scenario with people moving closer or further away from the camera, i.e., the foreground object size is varying over time. (c) The scene "trees" address the issue of quasi-static environments where a swaying tree is present as the dynamic background object. (d) The scene "parklot" presents a environment with walking people and moving cars of different size. Gradual illumination as well as "waking" foreground objects are also within the focus. Video sequences are provided by AXIS Communications [36].

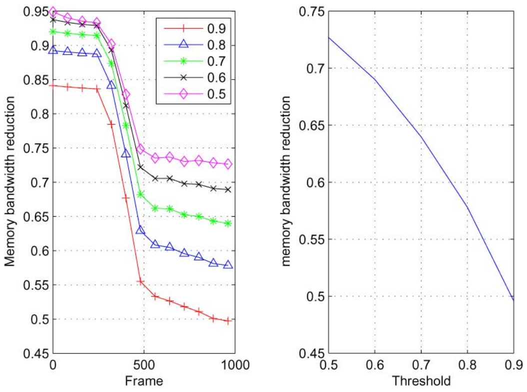


Fig. 9. Memory bandwidth reductions achieved in an indoor test video sequence. Left: memory bandwidth reductions with various threshold values are given over frames. It shows memory bandwidth savings tend to stabilize after initialization phase. Right: Stabilized memory bandwidth reductions against threshold value. Lower threshold value results in better bandwidth reductions at the cost of introducing more noises in the segmented image.

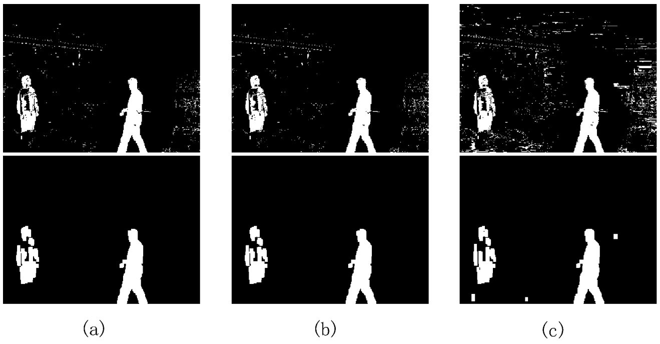


Fig. 10. Results before and after morphological filtering for different thresholds. (a) Original result. (b) Threshold of 0.8. (c) Threshold of 0.4.

distributions per pixel. With the proposed memory reduction schemes, off-chip memory bandwidth is reduced by more than 70%. A summary of the whole design is given in Table III.

In Table IV, a summary of the hardware complexities of different blocks is given. It can be seen that algorithm modifications results in a relatively low hardware complexity in parameter updating block, which merely occupies 8% of the total utilized resources. Furthermore, the hardware complexity of sorting and color space transformation is low after optimization. However, the memory reduction scheme with pixel locality is relatively costly in hardware where a relatively large number of multiplications are needed to calculate the cube volumes. However, for video application it is often advantageous to trade memory bandwidth for arithmetic complexity. The

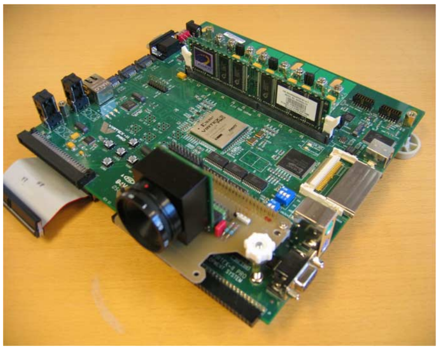


Fig. 11. Segmentation hardware is prototyped on an Xilinx VirtexII vp30 FPGA development board. A DDR memory is used for storing one frame of Gaussian parameters. A KODAK KAC-9648 CMOS sensor [39] is used to capture color image stream into the FPGA platform.

TABLE III
DESIGN SUMMARY

FPGA Utilization			
Nr. of Slices	Nr. of Flip Flops	Nr. of DCMs	Nr. of BRAMs
6107	4273	5	84
Clock Domains			
100Mhz		16Mhz	25Mhz
Sensor interface & DDR controller		Segmentation	VGA controller
System Parameters			
Resolution	Throughput	Frame rate	Nr. Gaussians
640 × 480	170MB/s	25	3

TABLE IV
HARDWARE COMPLEXITY FOR DIFFERENT BLOCKS WITHIN SEGMENTATION UNIT

Logic Block	Nr. of Slices (%)	Nr. Block RAMs
Match	253 (4%)	0
Switch	652 (11%)	0
Parameter Update	483 (8%)	0
Pixel Locality	1530 (25%)	0
Sensor Interface	540 (9%)	3
Sorting	355 (6%)	0
DDR controller	1599 (26%)	24
$RGB2YC_bC_r$	181 (3%)	0
VGA controller	377 (6%)	57

DDR controller contributes to a large part of the whole design due to complicated memory command and data signal manipulations, clock schemes, and buffer controls. Block RAMs are used as data buffers to support DDR burst read and write operations. The 24 BRAMs used for the DDR controller can be reduced by using low depth Gaussian parameter buffers to write/read to the off-chip DDR memory. However, the number of burst operations to DDR memory is increased which causes overhead. There is a tradeoff between DDR performance and on-chip buffer size and with different DDR parameters, the optimum buffer size would vary. A dedicated VGA controller is designed streaming output data into a monitor where the results from different stages of the logic can be viewed. The VGA controller consumes most of the on-chip memory resources, but is not really a part of the segmentation algorithm. Dual-port block RAMs are used as video RAMS in the VGA controller, which are shared by different blocks of the complete surveillance system to display the results from different stages on a monitor. Thus, the memory requirements directly dedicated to the algorithm is low while the DDR and VGA controller utilize a substantial amount of memory.

V. CONCLUSION

A real-time video segmentation unit is implemented on a Xilinx FPGA platform capable of 25 fps at VGA resolution. By utilizing combined memory reduction schemes, off-chip memory access can be reduced by over 70%. With real time performance, tracking schemes can be evaluated in varied environments for system robustness testing. For the implementation of the hardware units, memory usage is identified as the main bottleneck of the whole system, which is common in many image processing systems. This is especially true for segmentation since most algorithms operates in pixel-wise structures. To address the issue a joint memory reduction scheme is proposed by utilizing pixel locality and wordlength reduction. By measuring similarity of neighboring Gaussian distributions with overlapping volume of two cubes, threshold can be set to classify Gaussian similarities. Wordlength reduction is as important for memory bandwidth reduction. By utilizing coarse parameter updating scheme, wordlength for each Gaussian parameters are reduced substantially, which effectively decrease the memory bandwidth to off-chip memories. Careful tradeoffs should be made based on different application environments. Algorithm modifications are of great importance for the efficiency of the hardware implementation.

ACKNOWLEDGMENT

The authors would like to thank Xilinx for donating FPGA boards, and AXIS Communications AB for their network camera and expertise on image processing. The software simulations are performed based on the image data captured by AXIS network camera products.

REFERENCES

[1] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 1999, pp. 255–261.

[2] S. Chien, S. Ma, and L. Chen, "Efficient moving object segmentation algorithm using background registration technique," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 7, pp. 577–586, Jul. 2002.

[3] D. A. Migliore, M. Matteucci, and M. Naccari, "View-based detection and analysis of periodic motion," in *Proc. 4th ACM Int. Workshop Video Surveill. Sensor Netw.*, Oct. 2006, pp. 215–218.

[4] J. Moon, D. Kim, and R. Park, "Video matting based on background estimation," *Proc. World Acad. Sci., Eng. Technol.*, vol. 2, pp. 149–152, Jan. 2005.

[5] Q. Zhang and R. Klette, "Robust background subtraction and maintenance," in *Proc. 17th ICPR*, Cambridge, U.K., Aug. 2004, pp. 90–93.

[6] R. Cutler and L. Davis, "View-based detection and analysis of periodic motion," in *Proc. 14th Int. Conf. Pattern Recognit.*, Brisbane, Australia, Aug. 1998, pp. 495–500.

[7] B. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," in *Proc. Int. Symp. Intell. Multimedia, Video Speech Process.*, Hong Kong, May 2001, pp. 158–161.

[8] F. Porikli, "Multiplicative background-foreground estimation under uncontrolled illumination using intrinsic images," in *Proc. IEEE Workshop Motion Video Comput. (WACV/MOTION'05)*, Jan. 2005, pp. 20–25.

[9] S. Haykin, *Adaptive Filter Theory*, 4th ed. Upper Saddle River, NJ: Prentice-Hall, 2001.

[10] J. Zhong and S. Sclaroff, "Segmenting foreground objects from a dynamic textured background via a robust Kalman filter," in *Proc. ECCV*, 2003, pp. 44–50.

[11] S. Messelodi, C. M. Modena, N. Segata, and M. Zanin, "A Kalman filter based background updating algorithm robust to sharp illumination changes," in *Proc. 13th Int. Conf. Image Anal. Process.*, Cagliari, Italy, 2005, pp. 163–170.

[12] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and foreground detection using Kalman-filtering," in *Proc. ICRAM*, Istanbul, Turkey, 1995, pp. 193–199.

[13] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conf. Compu. Vis. Pattern Recognit.*, 1999, pp. 246–252.

[14] O. Javed, K. Shafique, and M. Shah, "A hierarchical approach to robust background subtraction using color and gradient information," in *Proc. Workshop Motion Video Comput.*, 2002, pp. 22–27.

[15] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, "Foreground object detection in changing background based on color co-occurrence statistics," in *Proc. 6th IEEE Workshop Appl. Comput. Vis.*, 2002, pp. 269–274.

[16] M. Harville, G. Gordon, and J. Woodfill, "Adaptive video background modeling using color and depth," in *Proc. IEEE Int. Conf. Image Process.*, 2001, pp. 90–93.

[17] Y. Zhang, Z. Liang, Z. Hou, H. Wang, and M. Tan, "An adaptive mixture Gaussian background model with online background reconstruction and adjustable foreground mergence time for motion segementation," in *Proc. IEEE Int. Conf. Industrial Technol.*, 2005, pp. 23–27.

[18] H. Wang and D. Suter, "A re-evaluation of mixture-of-Gaussian background modeling," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, 2005, pp. 1017–1020.

[19] Y. Xu, D. Xu, Y. Liu, and A. Wu, "Illumination invariant motion estimation for video segmentation," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, 2004, pp. 737–740.

[20] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proc. Eur. Conf. Comp. Vis. (ECCV)*, Dublin, Ireland, pp. 751–767.

[21] H. Jiang, H. Ardö, and V. Öwall, "Real-time video segmentation with VGA resolution and memory bandwidth reduction," in *IEEE Int. Conf. Video Signal Based Surveill.*, Sydney, Australia, Nov. 2006, pp. 104–109.

[22] N. Ratha and A. Jain, "Computer vision algorithms on reconfigurable logic arrays," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 1, pp. 29–43, Jan. 1999.

[23] W. MacLean, "An evaluation of the suitability of FPGAs for embedded vision systems," in *Proc. 2005 IEEE Comp. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 131–131.

[24] K. Shimizu and S. Hirai, "Realtime and robust motion tracking by matched filter on CMOS + FPGA vision system," in *Proc. 2007 IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 131–131.

[25] C. Cardon, W. Fife, D. Archibald, and J. K.Lee, "Fast 3-D reconstruction for small autonomous robots," in *Proc. 32nd Ann. Conf. IEEE Ind. Electron. Soc.*, 2005, pp. 373–378.

[26] P. Dillinger, J. Vogelbruch, J. F.Leinen, R. Suslov, S. Patzak, H. Winkler, and K. Schwan, "FPGA-based real-time image segmentation for medical systems and data processing," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 4, pp. 2097–2101, Aug. 2006.

[27] A. Darabiha, J. Rose, and J. Maclean, "Video-rate stereo depth measurement on programmable hardware," in *Proc. 2003 IEEE COMSOC Conf. Compu. Vis. Pattern Recognit.*, Jun. 2003, pp. 203–210.

[28] M. Magee, "Tracking multiple vehicles using foreground, background and motion models," *Image Vis. Comput.*, no. 22, pp. 143–145, 2004.

[29] F. Kristensen, P. Nilsson, and V. Öwall, "Background segmentation beyond RGB," in *Proc. Asian Conf. Comput. Vis.*, 2006, pp. 602–612.

[30] G. Szedo, "Color-space converter: RGB to YC_rC_b," 2008 [Online]. Available: http://www.xilinx.com/bvdocs/appnotes/xapp930.pdf

[31] F. Kristensen, H. Hedberg, H. Jiang, P. Nilsson, and V. Öwall, "Hardware aspects of a real-time surveillance system," in *Proc. Eur. Conf. Comp. Vis. (ECCV)*, 2006, pp. 161–168.

[32] F. Kristensen, H. Hedberg, H. Jiang, P. Nilsson, and V. Öwall, "An embedded real-time surveillance system: Implementation and evaluation," *J. VLSI Signal Process. Syst.*, to be published.

[33] K. E. Batcher, "Sorting networks and their applications," in *Proc. AFIPS Spring Joint Comput. Conf.*, 1968, pp. 307–314.

[34] M. Ajtai, J. Komlos, and E. Szemeredi, "An $O(n \log N)$ sorting network," in *Proc. 25th ACM Symp. Theory Comput.*, 1983, pp. 1–9.

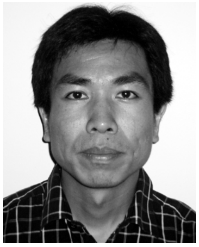
[35] G. V. Russo and M. Russo, "A novel class of sorting networks," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 43, no. 7, pp. 544–552, Jul. 1996.

[36] Axis Communications, 2008 [Online]. Available: http://www.axis.com/

[37] H. Hedberg, F. Kristensen, P. Nilsson, and V. Öwall, "Low complexity architecture for binary image erosion and dilation using structuring element decomposition," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, pp. 3431–3434.

[38] H. Hedberg, F. Kristensen, and V. Öwall, "Low-complexity binary morphology architectures with flat rectangular structure elements," *IEEE Trans. Circuits Syst.—Part I: Reg. Papers*, vol. 55, no. 8, pp. 2216–2225, Aug. 2008.

[39] Kodak Image Sensor (2008). [Online]. Available: http://www.kodak.com/US/en/dpq/site/SENSORS/name/ISSCMOSProductFamily



Hongtu Jiang received the M.Sc. degree from Jilin University, Jilin, China, and the Ph.D. degree from Lund University, Lund, Sweden, in 2000 and 2007, respectively, both in electrical engineering.

In 2007, he joined Ericsson Mobile Platform (EMP), Lund, Sweden, where he works as a Staff Engineer in the multimedia group in digital ASIC department. His main research interest is in the hardware implementations of image/video processing systems.

Håkan Ardö received the M.Sc. degree in electrical engineering from Lund University, Lund, Sweden, in 2002, where he is currently pursuing the Ph.D. degree in theMathematical Imaging Group. His thesis work consists of designing image processing algorithms for traffic surveillance, aiming for a system that automatically measures the safety of an intersection or road segment.

He worked as a software engineer at Axis Communications 2002–2003, and was then accepted as an Ph.D. student at the Centre for Mathematical Sciences of Lund University.

Viktor Öwall (M'90) received the M.Sc. and Ph.D. degrees in electrical engineering from Lund University, Lund, Sweden, in 1988 and 1994, respectively.

During 1995 to 1996, he joined the Electrical Engineering Department, the University of California at Los Angeles as a Postdoc where he mainly worked in the field of multimedia simulations. Since 1996, he has been with the Department of Electrical and Information Technology, Lund University, Lund, Sweden. His main research interest is in the field of digital hardware implementation, especially algorithms and architectures for wireless communication, image processing and biomedical applications. Current research projects include combining theoretical research with hardware implementation aspects in the areas of pacemakers, channel coding, video processing, and digital holography.

Dr. Öwall was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: ANALOG AND DIGITAL SIGNAL PROCESSING from 2000–2002 and is currently Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS.