Development of a New Tracking System Based on CMOS Vision Processor Hardware, Phase II Prototype Demonstration

Final Report

Prepared by:

Hua Tang
Peng Li

Department of Electrical and Computer Engineering
University of Minnesota Duluth

Northland Advanced Transportation Systems Research Laboratory
University of Minnesota Duluth

CTS 12-08

Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.	
CTS 12-08			
4. Title and Subtitle Development of a New Tracking System Based on CMOS Vision Processor Hardware, Phase II Prototype Demonstration		5. Report Date May 2012	
		6.	
7. Author(s) Hua Tang, Li Peng		8. Performing Organization Report No.	
9. Performing Organization Name and Address Department of Electrical and Computer Engineering University of Minnesota Duluth 1023 University Drive Duluth MN 55812		10. Project/Task/Work Unit No. CTS Project #2010008	
		11. Contract (C) or Grant (G) No.	
12. Sponsoring Organization Name and Address Intelligent Transportation Systems Institute Center for Transportation Studies University of Minnesota 200 Transportation and Safety Building 511 Washington Ave. SE Minneapolis, MN 55455		13. Type of Report and Period Covered Final Report	
		14. Sponsoring Agency Code	
15. Supplementary Notes http://www.its.umn.edu/Publications/ResearchReports/			

16. Abstract (Limit: 250 words)

Intelligent transportation systems depend on being able to track vehicle operations and collect accurate traffic data. This project targets a hardware-based video detection system for real-time vehicle detection. To allow real-time detection, customized hardware implementation of the system is targeted instead on the traditional computer-based implementation of the system. The system includes four main processing steps. First, a camera is used to capture images. Second, the captured images are segmented using the Mixture-of-Gaussian algorithm. Without sacrificing the segmentation accuracy, researchers modified the Mixture-of-Gaussian algorithm to allow more efficient and economical hardware implementation in terms of design overhead and hardware resources. Third, the segmentation regions are extracted and validated as the objects of interests. In the last step, the validation result will be wirelessly transmitted to a variable message sign, which displays necessary traffic information. Since the system design includes integration of diverse devices, the video design kit from Xilinx is used. Such a hardware-based vehicle detection system has been experimented tested with practical videos of traffic scene.

17. Document Analysis/Descriptors		18. Availability Statement	
Intelligent transportation systems, Cameras, Real time information, Wireless communication systems, Implementation		No restrictions. Document available from: National Technical Information Services, Alexandria, Virginia 22312	
19. Security Class (this report) Unclassified	20. Security Class (this page) Unclassified	21. No. of Pages 38	22. Price

Development of a New Tracking System Based on CMOS Vision Processor Hardware, Phase II Prototype Demonstration

Final Report

Prepared by:

Hua Tang
Peng Li

Department of Electrical and Computer Engineering
University of Minnesota Duluth

Northland Advanced Transportation Systems Research Laboratory
University of Minnesota Duluth

May 2012

Published by:

Intelligent Transportation Systems Institute
Center for Transportation Studies
University of Minnesota
200 Transportation and Safety Building
511 Washington Ave. S.E.
Minneapolis, MN 55455

Acknowledgments

The authors wish to acknowledge those who made this research possible. The study was funded by the Intelligent Transportation Systems (ITS) Institute, a program of the University of Minnesota's Center for Transportation Studies (CTS). Financial support was provided by the United States Department of Transportation's Research and Innovative Technologies Administration (RITA).

The project was also supported by the Northland Advanced Transportation Systems Research Laboratories (NATSRL), a cooperative research program of the Minnesota Department of Transportation, the ITS Institute, and the University of Minnesota Duluth College of Science and Engineering.

The authors would like to thank Dr. Eil Kwon, director of Northland Advanced Transportation Systems Research Laboratory (NATSRL) at University of Minnesota Duluth, for very valuable discussions and comments.

Authors would also like to thank the Minnesota Department of Transportation (MnDOT) and District I office for providing all traffic videos tested in this project.

Table of Contents

List of Figures

List of Tables

Executive Summary

It is well known that vehicle tracking processes are very computationally intensive. Traditionally, vehicle tracking algorithms have been implemented using software approaches. In other words, the vehicle tracking system is physically implemented in the general computers or micro-computers regardless of the underlying vehicle tracking algorithm. While the software approach provides flexibility, efficiency and short design cycle in terms of implementation and independency from the tracking algorithm, it does have a large computational delay, which causes low frame rate vehicle tracking. However, real-time vehicle tracking is highly desirable to improve not only tracking accuracy but also response time, in some ITS (Intelligent Transportation System) applications such as security monitoring and hazard warning. For this purpose, this project makes an attempt to design a hardware-based system for real-time vehicle detection, which is typically a required frontend in the complete tracking system. The proposed vehicle detection system captures pictures using a camera in real-time and then applies Mixtures of Gaussian (MoG) algorithm for vehicle segmentation.

MoG algorithm is a statistical modeling technique of the background that can be used to build a background model of the road in terms of intensity of either grayscale or color images to allow relatively robust and sensitive vehicle segmentation. Compared to many other background modeling and vehicle segmentation methods, MoG has several important advantages, such as background update at a controlled pace to incorporate latest illumination and weather changes and multi-mode modeling to cope with repeated camera shaking. The latter is particularly needed as in our experience there is constant camera shaking due to wind in practical camera-based vision systems. MoG can help suppress many false segmentations that would otherwise appear. Another important advantage of the MoG algorithm for vehicle segmentation is that it is a good tradeoff between its algorithm performance and complexity of hardware implementation. This is particularly sought in our projects as we are targeting a hardware-based implementation of the vehicle detection system and such an implementation should not cause an excessively long design cycle and use an unreasonable amount of hardware resources.

In our project, we first verify the MoG algorithm for vehicle detection using computer simulations (or the software approach). We run the MoG algorithm in MATLAB on a long sequence of images that are taken from a pre-recorded video. It was found that the MoG algorithm gives satisfactory vehicle segmentation results and is robust to camera shakings. In addition, another important finding is that the maximum rate of vehicle segmentation using MATLAB software in a Desktop computer (with Intel T4400 Dual-Core processor in our case) was only 0.2 frames-per-second (fps), which is very low for real-time vehicle detection for security monitoring purposes, such as collision warning. This is because the MoG algorithm for video segmentation application is computational intensive.

To meet the real-time requirement of high-frame-rate, high-resolution video segmentation tasks, we then target a hardware implementation of the MoG algorithm. Once the MoG algorithm was verified using computer simulations, we modified the MoG algorithm to tailor it to be suitable to hardware implementation. Without modification, the MoG algorithm would cause a long design cycle and use a significant amount of hardware resources. In the project, we modified the MoG algorithm in a way that it is much more amenable to hardware implementation than the original one without sacrificing much on the accuracy of vehicle segmentation.

With the modified MoG algorithm, we then proposed a System-on-a-Chip (SoC) architecture for implementation of the MoG-based image segmentation algorithm. We designed a hardware Intellectual Property (IP) for the MoG algorithm. Moreover, we integrated the hardware IP into a SoC architecture, so that some key parameters, such as learning rate and threshold, can be configured on-line, which makes the system flexible to adapt to different environments. The proposed system has been implemented and tested on Xilinx XtremeDSP Video Starter Kit Spartan-3ADSP 3400A Edition. Experiment results show that under a clock frequency of 25MHz, this design meets the real-time requirement for Video Graphics Array (VGA) resolution (640×480) at 30 frames-per-second (fps). The maximum rate for vehicle detection could reach 81fps. Our experiments show that the vehicle segmentation results from the proposed hardware implementation are very comparable to those from the software implementation that implements the original MoG algorithm.

Finally, in this project we incorporate the designed MoG-based vehicle segmentation system as the core of a complete vehicle detection system. In the complete vehicle detection system, the vehicle segmentation results are subsequently processed by a vehicle identification module, which is currently implemented in the SoC architecture based on a simple foreground-pixel counting based algorithm. Once a vehicle is indentified, SoC architecture sends the detection result to a variable message sign using a wireless transmitter. Finally, the variable message sign, which accepts the message through a wireless receiver, displays the detection result. In the prototype system, we use a computer monitor as a replacement of the variable message sign. We verify the prototype system in the laboratory using practical traffic videos. We also apply it to facilitate vehicle merging at ramps where vehicles cannot see each other in different ramps that merge to the same main road.

1 Introduction

Automatic real-time vehicle tracking in Intelligent Transportation System (ITS) applications require real-time vehicle detection as the pre-requisite. Among many existing technologies for real-time vehicle detection such as microwave and radar, the camera-based approach is getting more and more popular today, partially because cameras allows wide view coverage and the recorded video allows most comprehensive traffic data collection. In camera-based systems, real-time vehicle detection is equivalent to real-time image segmentation from image processing point of view.

In this project, we attempted to apply vehicle segmentation techniques for real-time vehicle detection and implemented the real-time vehicle detection system in hardware which is the main research target of Phase II of the project. In Phase I of the project [1], we attempted to use motion estimation techniques, such as Fixed Block Size Motion Estimation (FBSME), Reconfigurable Block Size Motion Estimation (RBSME), Variable Block Size Motion Estimation (VBSME) for vehicle segmentation and proposed the corresponding hardware implementations. However, in [1], while the vehicle detection results tested very promising, it did suffer from one practical issue, which is camera shaking caused by wind. Camera shaking causes the motion estimation to be relatively unstable, therefore the results are not reliable in those cases. For details, readers are referred to [1].

In phase II of the project, we need to address the camera shaking problem that was observed in Phase I. To this end, we proposed to use Mixtures of Gaussian (MoG) algorithm for robust vehicle segmentation under even constant camera shaking and also proposed an efficient hardware implementation of the MoG algorithm. We have also integrated the hardware design for the MoG-based vehicle segmentation into a complete vehicle detection system and experiments have validated the system functionality.

In the following, we first give a brief introduction to the MoG algorithm for video segmentation. Then, we discuss the previous work on hardware implementation of the MoG algorithm. Finally, we give a summary and organization of the rest of the report.

1.1 MoG

MoG is a statistical modeling technique of the road background and allows object (or the foreground) segmentation. The MoG algorithm for video segmentation is first proposed by Stauffer and Grimson [2]. In images of traffic scenes, each pixel is modeled probabilistically with a mixture of several Gaussian distributions (typically 3 to 5) and each background sample is used to update the Gaussian distributions, so that the history of pixel variations can be stored in the mixture of Gaussian distributions. It is clear that the recent variations would dominate the MoG model. In a similar way, latest light and weather changes can be reflected by updating the mean and variance of MoG. An illustration of MoG modeling is shown in Figure 1. In Figure 1, the same spot in the road has different background samples at different times and the history variations are all recorded in the MoG model. Note that the MoG model would record all samples of that particular spot. In other words, this means that the MoG model record not only the actual road background, but also the vehicles, grass, trees, clouds, rain, snow, sunshine and whatever that occurs on the spot.

Suppose that 5 Gaussian distributions are used for the MoG model of each pixel of an image. Each Gaussian distribution has three parameters associated with it, namely the mean, variance and weight. The former two are clear from mathematics point of view. The weight parameter of a Gaussian distribution is used to indicate its importance or dominance in the overall mixture of 5 Gaussian distributions. Typically, the sum of the weights of 5 Gaussian distributions would be set to unity (1.0). A few other important parameters include a threshold of *T* and a number *B*. The threshold is used to determine the fraction of the overall 5 Gaussian distributions that should account for the actual road background, because as mentioned above the MoG model records all samples including both desired/undesired background (undesired background samples for example are rain, snow, sunshine and so on) and foreground (vehicles). It is obvious that *T* is associated with the sum of weight and therefore *T* should be less than 1.0. A typically value of *T* is for example 0.75 and its optimal value for a particular traffic scene can be empirically and experimented evaluated. Once *T* is defined, *B* defines the number of Gaussian distributions that meet the threshold *T* or that belong to the distributions representing the background. In our example, there are a total of 5 Gaussian distributions, therefore *B* is less than 5.

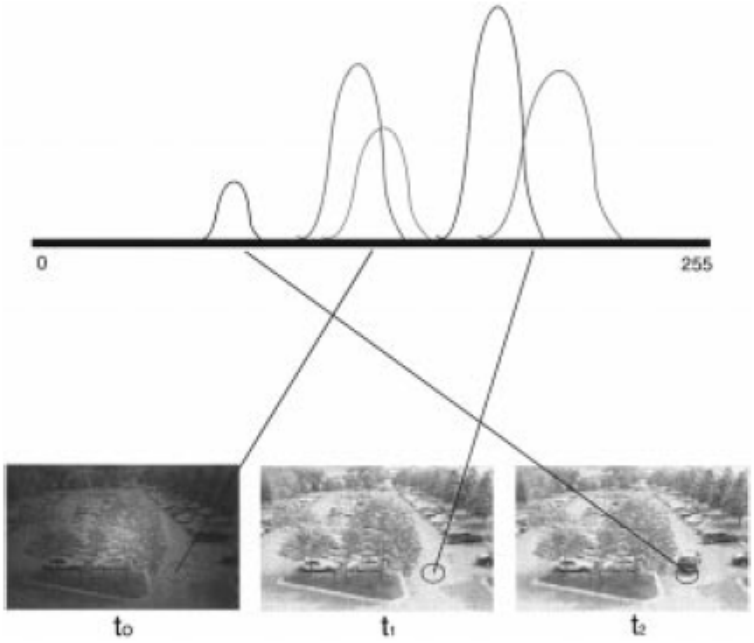


Figure 1.1. An illustration of Mixture of Gaussian for object segmentation [2].

When using the MoG algorithm for object (vehicles in this case) segmentation, each new sample (typically the intensity of a grayscale or a color image) of a certain pixel is matched with the current MoG model of that pixel. A match is defined as the new sample being within *J* times the standard deviation of the Gaussian away from the mean. If a match is found, two additional cases needs to be considered before determining whether the new sample represents background or foreground. In the above, *B* Gaussian distributions belong to the background ones, while the rest (5-*B*) to the foreground ones. Therefore, if the new sample matches any distribution belonging to the background ones, the sample is a background sample. On the other hand, if it matches any distribution belong to the foreground ones, it is a foreground sample. In each of above two cases, the MoG model would be updated and more specifically the matched Gaussian

distribution would be updated with a new mean, variance and weight. Especially important is the weight update and the weight of the matched Gaussian distribution will be weighted up since there is one more sample now. For the other 4 Gaussian distributions, theirs means and variance would not change, but their weights would be reduced by a quarter of the weight increment of the matched Gaussian distribution in order to sum up all the weight still equal to 1.0.

If the new sample does not match any of 5 Gaussian distributions, it would be declared as a foreground sample. In that case, the MoG model update would be slightly different from the above. Among the 5 Gaussian distributions, the one with the smallest weight would be deleted and be replaced by a new Gaussian distribution associated with the new sample. More specifically, the new Gaussian distribution would have a mean equal to the sample, an initialized variance and its weight equal to that of the deleted or the replaced one.

In summary, the MoG algorithm can provide multi-mode modeling of the background (if $B \geq 1$) and in this way of modeling the background, the sensitivity and accuracy of vehicle detection is highly improved compared to traditional techniques. In Chapter 2, we will give more detailed mathematical description of the algorithm and discuss its benefits and motivates why it was chosen in our project.

1.2 Hardware implementation of MoG

In the above subsection, we give a brief introduction of the MoG algorithm to be used as the vehicle segmentation algorithm. As implementing the MoG-based vehicle segmentation in hardware to allow real-time vehicle detection is the main goal of the project, we also review briefly some previous and related work on hardware implementation of vehicle segmentation algorithm in this subsection.

Conventional image segmentation techniques include non-adaptive methods such as background subtraction and adaptive methods such as frame difference (FD) [3-6]. The non-adaptive methods have almost been abandoned because their need for manual initialization. Without re-initialization, background noise accumulates over time. In another word, they are not suitable for highly automated surveillance environments or applications. Besides FD, there are several other adaptive video segmentation methods, such as median filter (MF) [7-9], linear predictive filter (LPF) [10-13], MoG [2][14-19], and Kernel Density Estimation (KDE) [20].

When selecting the algorithm for image segmentation that needs to be implemented in hardware, a few factors come into play. A comparison among these methods has been made by Jiang et al. [21] in terms of performance, memory requirement, segmentation quality, and hardware complexity. We cite their results in Table 1.1. It can be seen that MoG has arguably the best trade-off among these adaptive segmentation algorithms.

Although sophisticated video segmentation algorithm development is a hot topic in the research community, only a few research groups work on the hardware implementation of the algorithms to meet real-time requirement of high-frame-rate, high-resolution video segmentation tasks. In our software implementation of a 3-mixture MoG algorithm (in MATLAB) on an Intel T4400 Dual-Core processor, the performance is about 5 second per frame with the VGA resolution (640×480), which cannot even meet the real-time requirement of 1fps.

The latest work about hardware implementation of the MoG algorithm so fast as we know is from Jiang et al. [21]. In [21], the MoG algorithm is translated into hardware and implemented in Xilinx VirtexII pro Vp30 Field Programmable Gate Array (FPGA) platform [22]. It can support VGA resolution (640×480) at 25 fps in real-time. The authors also present a variety of memory access reduction schemes, which results in more than 70% memory bandwidth reduction. As a result, their design meets the real-time requirement of high-frame-rate high-resolution segmentation task with a relatively low hardware complexity.

Table 1.1. Comparison of different adaptive video segmentation algorithms [21].

	FD	**MF**	**LPF**	**MoG**	**KDE**
Algorithm Performance	Fast	Fast	Medium	Medium	Slow
Memory Requirement	1 frame	50-30 frames	1 frame	1 frame of k Gaussians	n frames of k Gaussians
Segmentation Quality	Worst	Low	Acceptable	Good	Best
Hardware Complexity	Very low	Medium	Low to Medium	Low	High

However, the design in [21] lacks flexibility. Some key parameters in the MoG algorithm, such as learning rate and threshold, have to be setup before generating the FPGA bit stream file. This means that every time if the users want to change the parameters, they have to re-program the FPGA, which is a time consuming process. Moreover, some commonly used components in embedded system, such as Universal Asynchronous Receiver/Transmitter (UART) and Inter-Integrated Circuit (I^2C), are not involved in their design, which also limits the application of their system.

As we know, video segmentation usually plays a preprocessing role for other applications, such as video surveillance, object detection, and object tracking. Thus, implementing the video segmentation algorithm into a hardware IP will be more desirable than implementing it into an entire system such as in [21]. The main reason is that a hardware IP can be easily integrated into an SoC architecture so long as it meets the specified bus standard. The SoC architecture has many advantages. First of all, it is easier to be updated for other applications. We can design other hardware IPs for applications based on the same bus standard and integrate them into the same system. Second, based on the SoC architecture, the parameter configuration task will be easier, which can be performed on-line via Micro Control Unit (MCU) in the SoC. Third, some commonly used components in embedded system, such as UART and I^2C, can be easily integrated into the SoC. In summary, the SoC architecture makes the overall system more flexible.

Based on the above discussion, we implemented the MoG algorithm into a hardware IP, and then integrated it into an SoC architecture in this project. Similar to [21], we made some modifications to the original MoG algorithm in order to reduce the hardware design complexity and save hardware resources. However, we did not modify the original algorithm in the same way as [21]. This is mainly because the Xilinx FPGA in our platform has more resources than

the one in [21]. As a result, some algorithm modifications which will degrade segmentation performance are not used in our design. In another word, the modified MoG algorithm in our system is closer to the original MoG algorithm than the one in [21].

1.3 Summary

In this Chapter, we give a brief introduction of the MoG algorithm for video segmentation and its hardware implementation. The rest of this report is organized as follows:

- Chapter 2 presents the MoG algorithm and its benefits in detail and especially how it can cope with repeated camera shaking.
- Chapter 3 presents the SoC architecture for video segmentation based on the MoG algorithm. It includes a modified version of the MoG algorithm for hardware implementation, the structure of the MoG hardware IP, the overall SoC architecture, a prototype vehicle detection system based on the vehicle segmentation hardware and some experimental results.
- Chapter 4 presents the summary, conclusion and discussion of the works in the project.

2 Mixture of Gaussian for Video Segmentation

In this chapter, we will explain why we would prefer the MoG algorithm to the previous motion estimation algorithm used in Phase I of the project [1]. We first provide a mathematical description of the MoG algorithm, which is needed to help understand the computation complexity of the algorithm and why it should be modified for efficient hardware implementation and how in Chapter 3. Then, we present in detail the advantage of the MoG algorithm, especially its capability to cope with repeated camera shaking so that much less false vehicle segmentations would occur.

2.1 The MoG algorithm

The MoG algorithm for video segmentation is first proposed by Stauffer and Grimson [2]. This algorithm considers the values of a pixel at a particular position (x, y) of an image over time t as a "pixel process", and the recent history of the pixel is modeled by a mixture of K Gaussian distributions. The probability of observing the pixel is

$$P(X_t) = \sum_{i=1}^{K} w_{i,t}\eta(X_t, \mu_{i,t}\Sigma_{i,t}) \tag{3.1}$$

where X_t stands for the incoming pixel at time t or frame t (note that X_t would be the intensity in a gray scale image or the intensity of a certain color, such as Red, Green or Blue in a color image), K is the number of Gaussian distributions, and η is a Gaussian probability density function. Moreover, $w_{i,t}$ is the weighting factor, $\mu_{i,t}$ is the mean value and $\Sigma_{i,t}$ is the covariance matrix of the i^{th} Gaussian at time t.

When a new frame of image comes, each pixel sample will be considered whether it belongs to any existing distribution of that pixel. This procedure is called matching. There are some ways to match a pixel with Gaussian Distributions. [23] uses the Jeffrey's divergence which measures how unlikely that one distribution is drawn from the population represented by the other. The authors claim this method yield accurate results. One disadvantage of this method is its complexity. In our project, we applied the typical method, using confidence intervals, for matching. In statistics, a confidence interval of a Gaussian distribution is a particular kind of interval estimate of a population parameter. Instead of estimating the parameter by a single value, an interval likely to include the parameter is given, which indicates the reliability of an estimate. We choose confidence interval to be 98%, so that an incoming pixel sample X will match one distribution $D(\mu,\sigma)$ if:

$$p = \frac{|X - \mu|}{\sigma} \leq J = 2.5$$

In other words, a match is defined as the incoming pixel within J (usually set to 2.5) times the standard deviation off the center. In case that there are more than one distribution matching the sample, the distribution which has smallest p will be chosen for updating.

After matching, the next important step is updating the MoG model. The updating is an important part to keep the background model flexible with environment changes such as light, sunshine, scene changes, etc. This procedure will:

- Create a new Gaussian distribution in the MoG model if there is no match.
- Increase the weight of the Gaussian distribution that matches the incoming pixel sample, and then update the mean and standard deviation of that distribution. Meanwhile, decrease the weights of other Gaussian distributions that do not match while keeping the total weight equal to 1.0.

If a match is found, for that matching Gaussian distribution, its w_i, μ_i, and δ_i ($\Sigma_{i,t} = \delta_i{}^2 I$) are updated as follows,

$$w_{i,t+1} = (1-\alpha)w_{i,t} + \alpha$$

$$= (1-w_{i,t})\alpha + \alpha \tag{3.2}$$

$$\mu_{i,t+1} = (1-\beta)\mu_{i,t} + \beta X_t \tag{3.3}$$

$$\delta_{i,t+1}{}^2 = (1-\beta)\delta_{i,t}{}^2 + \beta(X_t - \mu_{i,t})^T(X_t - \mu_{i,t}) \tag{3.4}$$

where α is defined as the learning rate, which essentially indicates how fast the MoG model should learn new samples [2]. When $w_{i,t}$ is large, the amount (1- $w_{i,t}$) is small and $w_{i,t+1}$ will increase less. This assures that no distribution is likely to dominate the whole MoG model and small-weight distributions can adapt faster in the MoG model if it matches. β is the second learning rate, which is defined as follows

$$\beta = \alpha \times \frac{1}{\sqrt{2\pi\delta_{i,t}{}^2}} \times e^{\frac{(X_t-\mu_{i,t})}{2\delta_{i,t}{}^2}}$$

It can be seen that β involves significant computations, which will take large amount of hardware resources if directly implemented in hardware. Therefore, in Chapter 3, we modified the definition of β so its computation is much easier as long as the modification does not significantly sacrifice the algorithm accuracy.

On the other hand, when a match is found, for those distributions that do not match, then the following equations are used to update the weights, means, and variances as follows

$$w_{j,t+1} = (1-\alpha)w_{j,t} \tag{3.5}$$

$$\mu_{j,t+1} = \mu_{j,t} \tag{3.6}$$

$$\delta_{j,t+1}{}^2 = \delta_{j,t}{}^2 \tag{3.7}$$

$$for\ j \neq i$$

Basically, for those distributions, their weights are reduced while their means and variances are kept the same. Please note that the sum of weights of all Gaussian distributions still equal to 1.0 after updates from equations (3.2) and (3.7).

If the new pixel sample does not match any of the distributions in the current MoG model of that pixel, then a new distribution with mean equal to the sample value and an initialized standard deviation is created. The weight of the new distribution will be that of the lowest-weight distribution currently in the MoG model. Also, this new distribution will replace that lowest-weight distribution such that the number of Gaussian distribution remains constant. The update can be described as follows

$$w_{new,t+1} = \min(w_{1,t}, w_{2,t}, \dots, w_{K,t}) \tag{3.8}$$

$$\mu_{new,t+1} = X_t \tag{3.9}$$

$$\delta_{new,t+1}{}^2 = 25 \tag{3.10}$$

where *K* stands for the total number of Gaussian distributions, say 3 or 5. Usually, all the Gaussian distributions in the MoG model are assorted in an ascending order, such that $w_{1,t}$ in equation (3.8) is the lowest weight. Therefore, the new distribution created with weight $w_{new,t+1}$ would then become $w_{1,t}$. Also note in equation (3.10), the empirical initial variance is set to 25.

As introduced in Chapter 1, a portion of the Gaussian distributions models the background and the rest models the foreground. The portion of the Gaussian distributions belonging to the background is defined to be

$$B = argmin_b(\sum_1^b w_i > T) \tag{3.11}$$

where *T*, the threshold, is a measure of the minimum portion of the data that could be accounted for by the background. Once the MoG model is properly updated, the incoming pixel sample is finally determined as either background or foreground. One case is that there is a match and the other case is not. In the first case the pixel sample could be declared as background or foreground and it depends on the Gaussian distribution that has a match with the pixel sample. If the Gaussian distribution belongs to the background distributions as mentioned above depending on threshold *T* and *B*, then the pixel sample is declared as background. On the other hand, if the Gaussian distribution belongs to the foreground distributions, then it is foreground. In the other case that there is no match, the pixel sample is declared as foreground.

2.2 The benefits of the MoG algorithm

There are several main benefits of the MoG algorithm compared to the other video segmentations algorithms including the motion estimation algorithm that was previously used in Phase I of the project.

First, the MoG model could be very responsive to fast illumination changes. Depending on the learn rate, the response rate could be adjusted. Compared to a fixed background model, this is clearly an advantage of the MoG model in that the background is kept up-to-date, which should help reduce false detections.

Second, the multiple Gaussian distributions of the MoG algorithm allow multi-mode modeling of the background. Typically, the number of Gaussian distributions could be between 3 and 6.

Suppose there is only a mixture of 3 Gaussian distributions used in a MoG model. Then, 2 of the 3 distributions could be used for background while the other for foreground. This means that a two-mode background could be accommodated, which is especially advantageous for repeated shaking or motion of the background. For instance, for the traffic scene captured by a surveillance camera as shown in Figure 2.1(a), in practice the camera keeps shaking subject to wind and therefore the image is not stable at all. The effect is that for the same spot in the image, sometimes it is the true road background while it is the grass at other times, especially along the road and grass borders. If we used the traditional background subtraction or frame difference method for vehicle segmentation, it is then highly likely that many false segmentations would occur. However, using the MoG algorithm, the two modes can be modeled by two Gaussian distributions, which may both belong to the background distributions. So in the MoG algorithm, false segmentations would be much reduced since the algorithm "understands" that a new pixel sample which originates from either the true road background or the grass can be both considered as background. This is a significant advantage when there is repeated camera shaking, which is always observed in all videos we received from state and local transportation departments. Figure 2.1(b) show the number of Gaussian distributions used to model the background. At each pixel, whiter means more are used to model the background and darker means less out of a total of 5 Gaussian distributions.

Figure 2.2 gives another example that the background has waving trees. Figure 2.2(a) shows the image, and (b) shows the number of Gaussian distributions used to model the background pixel-wise.

It is clear from the two examples that MoG algorithm can model the background in multiple modes. The multiple scene changes at each pixel caused by camera shaking or background waving (such as tree leaves, rain, snow, lake/sea surfaces etc) are recorded in the background distributions and are therefore not identified as false segmentation regions any more.

We show some results from vehicle segmentation. Figure 2.3(a) shows the image frame that has encountered significant camera shaking with respect to the previous image frame, and Figure 2.3(b) shows segmentation results in traditional background subtraction methods, and finally Figure 2.3(c) gives segmentation results with the MoG algorithm. It is noted the MoG algorithm are robust to false detections from camera shaking in comparison to other methods. False detections are much reduced and most remaining noise in Figure 2.3(c) is salt and pepper noise, which can be easily removed using median filters.

In our study and practical experiments in Phase I of the project, it was observed that camera shaking is a very realistic issue that cannot be neglected and the motion estimation algorithm for vehicle segmentation used in Phase I suffers from this problem, though it gives satisfactory results when there is little camera jitter. Therefore, in this phase we experimented with the MoG algorithm of statistical nature and it is shown in this Chapter that it gives satisfactory results even in case of repeated camera shaking.

(a) (b)

Figure 2.1. (a) A partial roundabout traffic scene captured by a nearby surveillance camera; (b) Number of Gaussian distributions used to model the background pixel-wise (whiter means more and darker less).

(a) (b)

Figure 2.2. (a) A scene with waving trees; (b) Number of Gaussian distributions used to model the background pixel-wise (whiter means more and darker less).

2.3 Summary

In this Chapter, we provide a detailed mathematical description of the MoG algorithm for video segmentation. Then, we discuss the advantages of the MoG algorithm to allow fast update and cope with repeated camera shaking. Experiment results have shown that the MoG algorithm is effective to reduce false segmentations resulting from camera shakings.

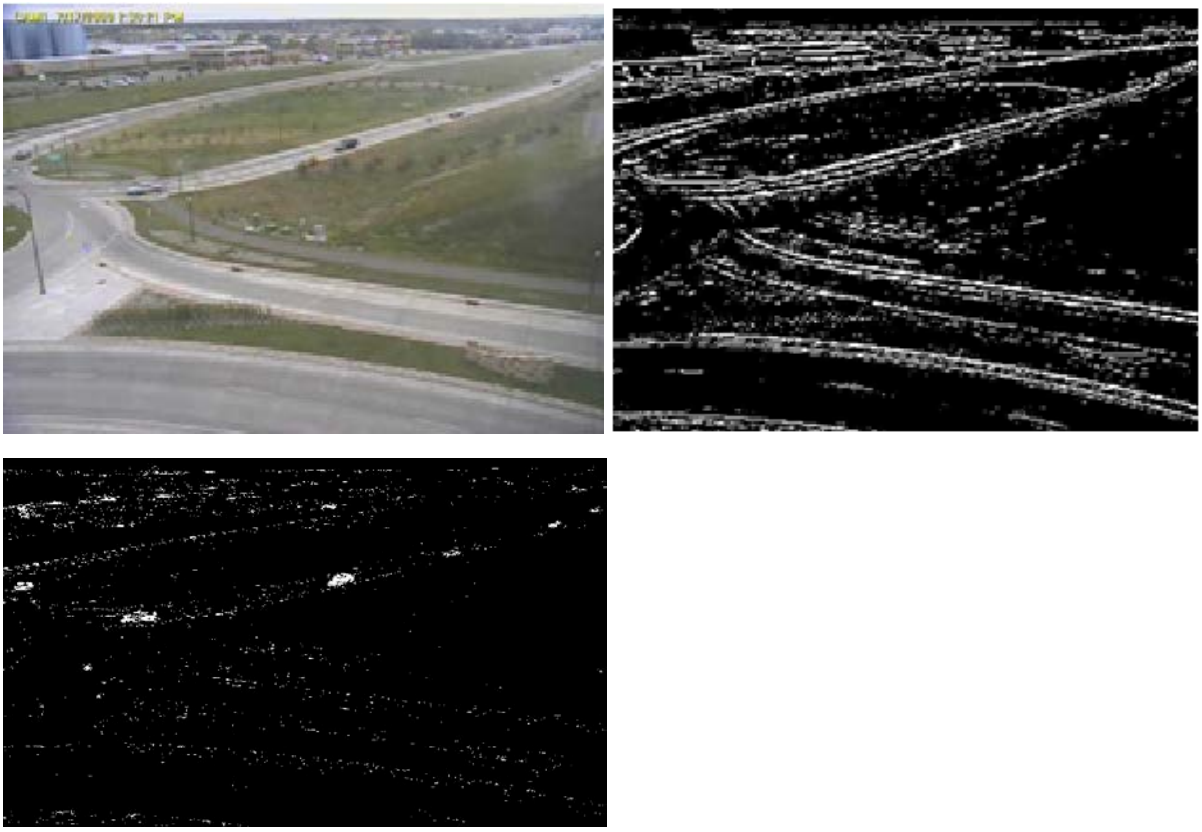


Figure 2.3. (a) The original image; (b) Segmentation results using traditional background subtraction methods; (c) Segmentation results using the MoG algorithm.

3 Hardware Implementation of MoG

In Chapter 2, we gave a detailed mathematical description of the MoG algorithm for vehicle segmentation and discussed its benefits, especially its robustness to camera shaking. However, blindly implementing the original MoG algorithm would takes unreasonable amount of design time and hardware resources. In this Chapter, we first modify the MoG algorithm to make it more amenable to hardware implementation considering the design cycle and the amount of hardware resources. When modifying the MoG algorithm, it was kept in mind that a good tradeoff between computation complexity and accuracy is the goal. As discussed in Chapter 1, the MoG algorithm does seem to provide the qualitatively best tradeoff among several video segmentation algorithms. After modification, we then present the hardware implementation of each major processing step of the modified MoG algorithm in detail, followed by the SoC implementation of the vehicle segmentation system. A prototype vehicle detection system built upon the MoG algorithm based vehicle segmentation system is also designed. Finally, experiment results of the vehicle detection system are shown.

3.1 Modification of the MoG algorithm

To translate the MoG algorithm into hardware with reasonable amount of design cycle and hardware resources, it is necessary to do some modification of the original algorithm. The modified algorithm should have almost the same segmentation performance as the original one, but be more suitable to hardware implementation. A pseudocode of the modified algorithm is show as follows,

========== Step 1.==========

for $i = 1\ to\ K$
$t(i) = (\mu_i - X_t)^2 - J^2\delta_i{}^2$;

========== Step 2.==========

$Find\ the\ minimum\ t(i), define\ r = \arg\min[t(i)]$;
$if\ (t(r) > 0)$
$match = 0\ (no\ match)$;
$else$
$match = 1\ (match)$;

========== Step 3.==========

$if\ (match = 0)$
$\mu_1 = X_t;\ \delta_1{}^2 = 25$;
$else$

$\mu_r = \mu_r + \beta(\mu_r - X_t)$;

$\delta_r{}^2 = \delta_r{}^2 + \beta[(\mu_r - X_t)^2 - \delta_r{}^2]$;

$w_r = w_r + \alpha(1 - w_r)$;

for $i = 1$ *to* K

$if\ (i\ !=r)$
$w_i = w_i - \alpha\ w_i$;

========== Step 4.==========

$if\ (match = 0)$
current pixel is foreground;
$else\ if\ [r < (K - B)]$
current pixel is foreground;
$else$
current pixel is background;

========== Step 5.==========

$Sort\ w_i\ in\ ascending\ order$;
$B = 0; Sum_w = 0; i = K - 1;$
$While\ (Sum_w < T)$
$Sum_w\ +=\ w_{i--}$;
$B++$;

Two main notes on the modified MoG algorithm are as follows:

- First, note that the second learning rate in step 3, β, as defined in Chapter 2.1, is simply taken as a constant in order to save hardware sources.
- Second, all the other computations of the original MoG algorithm are retained in order not to sacrifice accuracy, except that they are re-formulated for efficiency of hardware implementation. For example, a division operation for matching using confidence intervals (as defined in Chapter 2.1) in step 1 and 2 is preferred to be replaced by multiplication and subtraction to save design cycle and hardware resources. Compared to the hardware modification in [21] with simplified the parameter updates in step 3, the proposed modified version is more close to the original MoG algorithm and less sacrifice on accuracy is expected.

3.2 Hardware implementation of the MoG algorithm

After proper modification of the original MoG algorithm, the modified version can now be used for hardware implementation. We show the detailed hardware implementation of each major processing step of the modified algorithm in this Section.

The proposed hardware implementation of the modified MoG algorithm with three mixtures for a grayscale image is shown in Figure 3.1 (we will explain why we use three mixtures and a grayscale image in Section 3.3). The registers J^2, α, β, and T are used to store the global algorithm parameters, and the blocks labeled *Step1* to *Step5* in Figure 3.1 are the hardware implementations of the modified MoG algorithm presented in the previous subsection. Their

detailed circuits can be found in Figure 3.2 to Figure 3.6 respectively. Note that in the captions of each Figure, the number of pipelines means the number of clock cycles the circuit takes to get the output from the input. The corresponding pseudocode presented in the previous section explains what computations these circuits do.

Such a hardware implementation of the modified MoG algorithm can be then used as an IP and be integrated in an SoC hardware platform to allow build-up of a complete system, such as a vehicle detection system or a vehicle tracking system. This is discussed in the next Section.

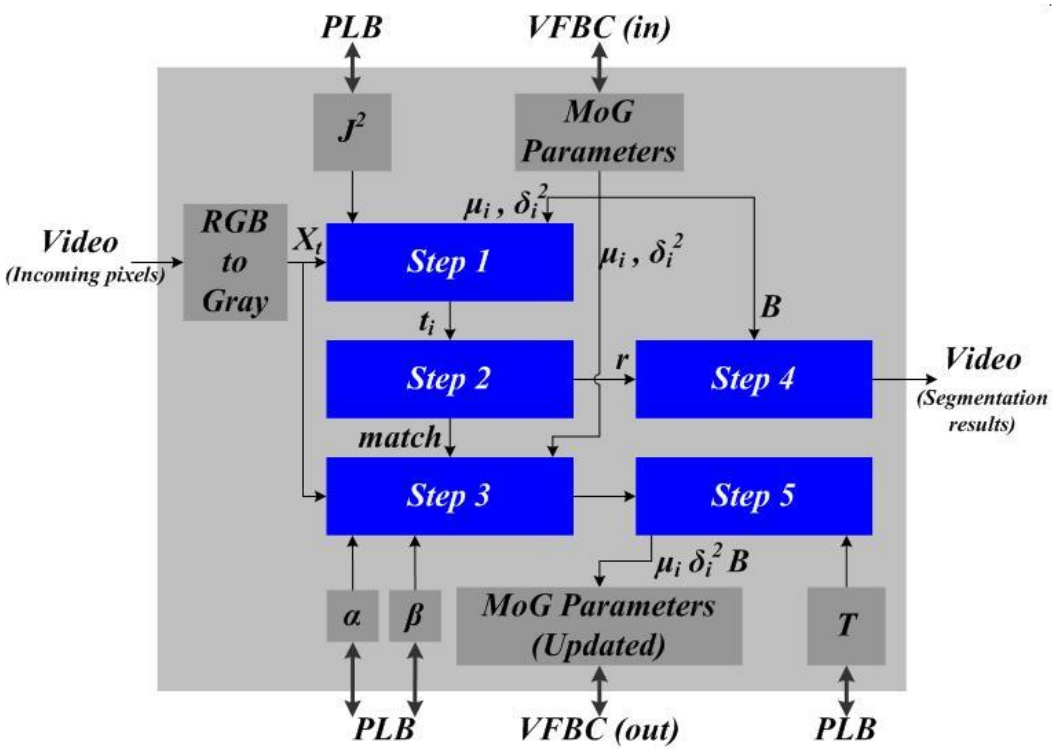


Figure 3.1. Overall architecture of the MoG IP.

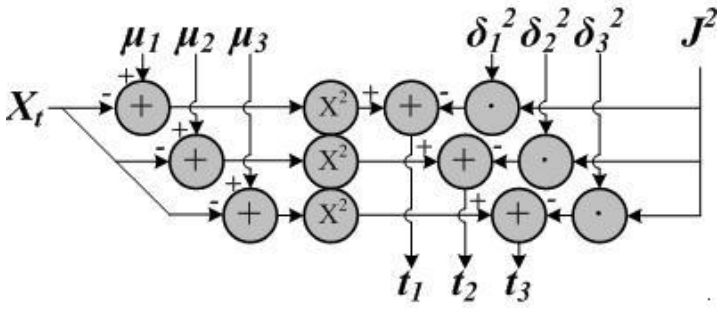


Figure 3.2. Circuits in step 1 (3-stage pipeline).

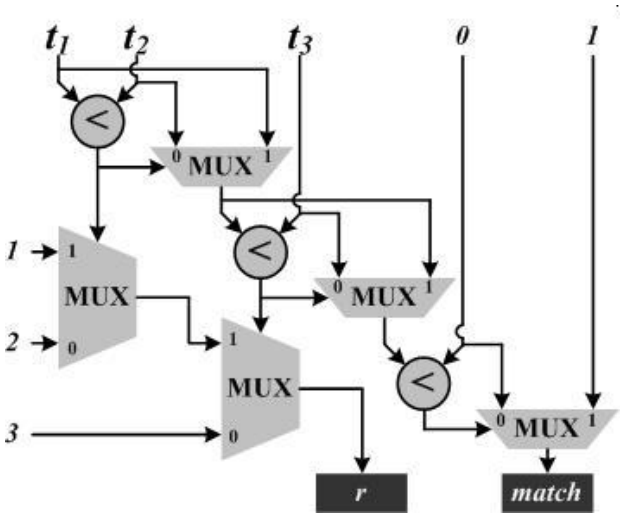


Figure 3.3. Circuits in step 2 (6-stage pipeline).

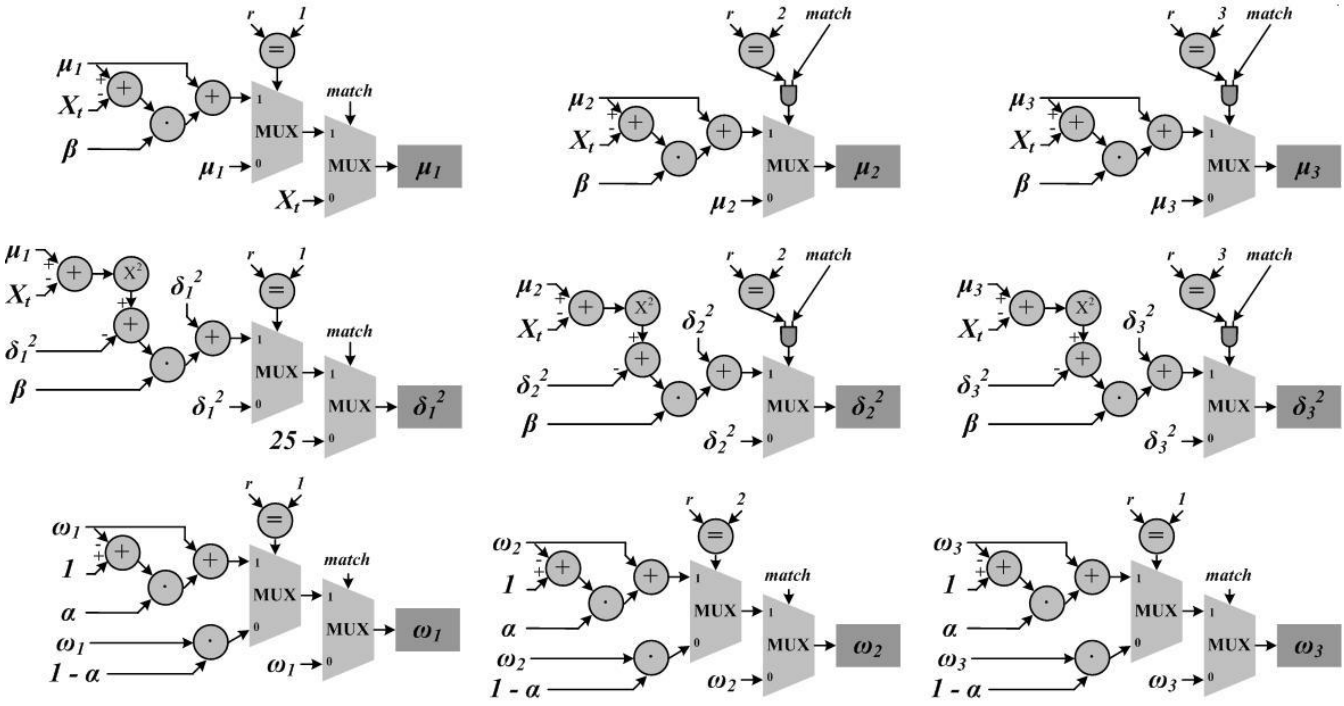


Figure 3.4. Circuits in step 3 (5-stage pipeline).

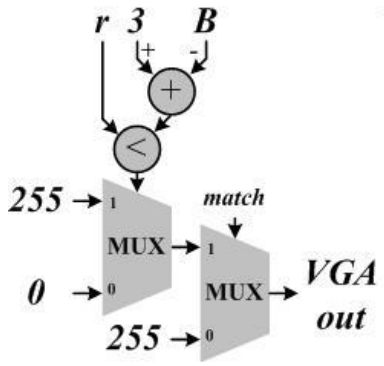


Figure 3.5. Circuits in step 4 (4-stage pipeline).

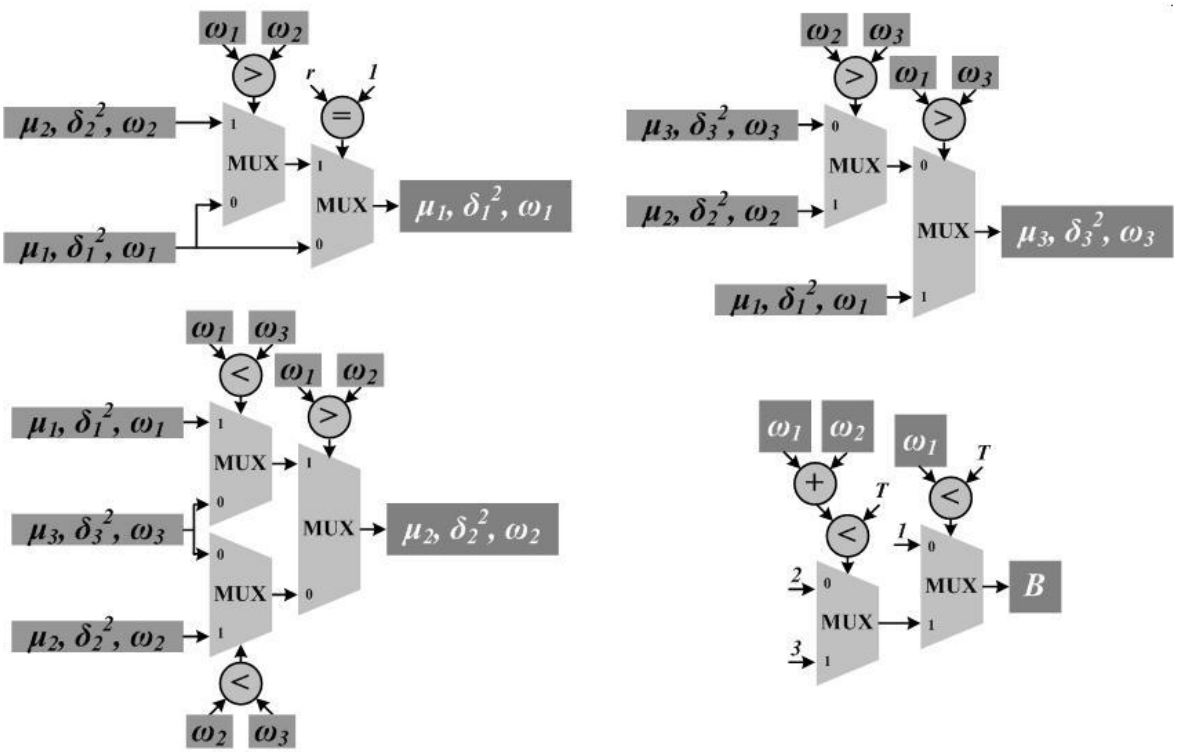


Figure 3.6. Circuits in step 5 (3-stage pipeline).

3.3 The SoC architecture

To integrate designed MoG IP into an SoC architecture, we need to use standard interfaces of the SoC platform. Since our hardware platform is based on Xilinx Spartan3A-DSP FPGA, we need to use three groups of interfaces. The first one is the video bus interface [22], which are used for incoming pixels and segmentation results (refer to Figure 3.1). The second one is the Processor Local Bus (PLB) interface [24], which connects the registers J^2, α, β, and T (as shown in Figure 3.1) to the Xilinx MCU MicroBlaze [25]. The third one is the Video Frame Buffer Controller (VFBC) bus interface [26], which connects the MoG IP to Xilinx multi-port memory controller

(MPMC) [26] to read the model parameters (VFBC In in Figure 3.1) and update the model parameters (VFBC Out in Figure 3.1). The overall SoC architecture is shown in Figure 3.7. All the IPs except the designed MoG IP are Xilinx's products, and are available from Xilinx. For more detailed information about these IPs, please see [22].

We use 8 bits for each register in the MoG IP. The bit width of the VFBC bus is set to 64, which is the maximum bit width of the bus. These 64 bits are assigned for the parameters of a MoG model of each pixel as follows. 8 bits are for the mean parameter μ of each mixture in the model, and 8 bits for the variance parameter δ^2 of each mixture in the model. All of these bits are used for integer part and no fraction part is used for the mean parameter or the variance parameter. 4 bits are for the weight parameter w of each mixture in the model. All of these bits are used for fraction part and no integer part is used for the weight parameter. As a result, for the three-mixture Gaussian model, $3 \times (8 + 8 + 4) = 60$ bits are used to store the mean, variance, and weight parameters. The last 4 bits of the VFBC bus are used for the parameter *B*. This explains that why we have used a three-mixture Gaussian model for a grayscale image as mentioned in Section 3.2. Note, if we used more mixtures or a color image, then the number of required bandwidth (or total number of bits to store the MoG parameters) would significantly increase and to accommodate it takes the VFBC bus more than one clock cycle to read in and store the MoG parameters, which would then slow down the processing of the MoG IP and eventually reduce the frame rate of vehicle segmentation. For example, using 5 mixtures for a color image (with Red, Green and Blue), the total number of bits is $3 \times 5 \times (8 + 8) + 5 \times 4 + 4 = 264$, which needs at least 5 clock cycles to read in and store them.

Next, we compute the required memory space to store all MoG parameters for a three-mixture Gaussian model and a grayscale image. For a typical 640×480 image, the required memory space is then 640×480×64 =2.35 Mega Bytes memory.

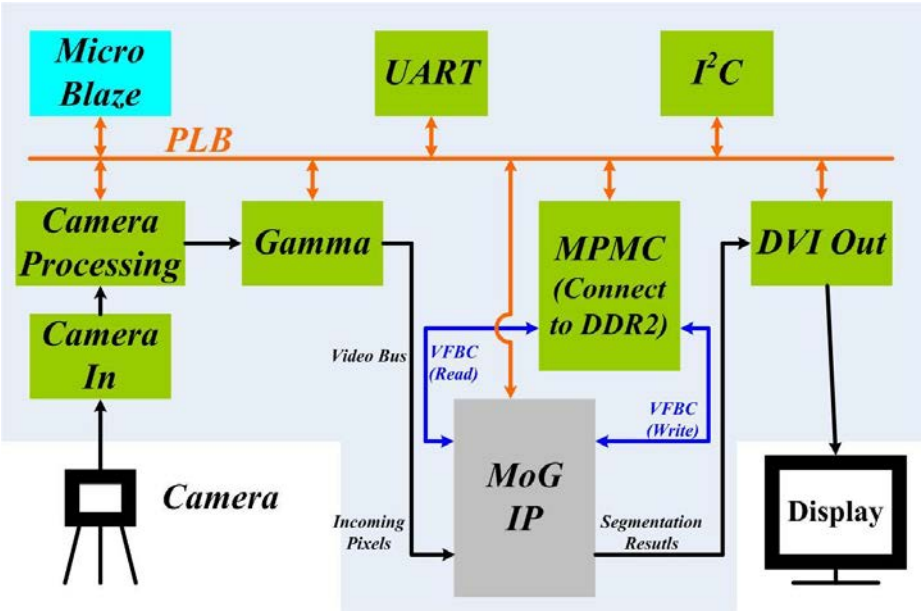


Figure 3.7. SoC architecture of the proposed design.

The proposed SoC design works under three clock domains. The components connected to PLB work under the 62.5MHz clock domain. The components connected to Video Bus and VFBC work under the 25MHz clock domain. The DDR2 SRAM (not shown in Figure 3.7) connected to MPMC work under the 125MHz clock domain. Some components connected both to PLB and Video Bus, such as Camera Processing and the MoG IP, have two clock domains. MPMC is the only component which works under all of the three clock domains.

The main advantage of this SoC architecture is that most of the components are connected to MicroBlaze via PLB. These components can communicate with each other via MicroBlaze, which makes the system very flexible. For example, if we want to control or reset the learning rate α of the MoG IP via UART, we can send commands to UART first. Then UART will send these commands to MicroBlaze via PLB. Once MicroBlaze gets these commands, it will configure the value of register α to the desired one via PLB. This process is independent from the video processing, which means that the configuration process can be performed on-line. In the MoG IP, the configurable parameters are J^2, learning rate α and β, and the threshold T. Users can configure these parameters on-line to most ideal values by an experiment approach. This is also a significant advantage compared to the implementation in [21] as it lacks flexibility of parameter control.

3.4 Experiment and performance analysis

The proposed design has been implemented and verified in Xilinx Spartan-3A DSP FPGA Video Starter Kit [22]. Design report from Xilinx ISE shows that the critical path delay of the proposed design is 7.766ns. The overall design takes 14k Slices, 12k Slice Flip Flops, 15k 4-Input LUTs, 77 BRAMs, 11 DSP48As, and 2 DCMs. The FPGA design summary is shown in Table 3.1. The hardware complexity of different blocks in the system is shown in Table 3.2. The system's parameters are shown in Table 3.3. This system can process one pixel per clock cycle due to the pipeline structure. If the video's resolution is 640×480, it can support up to

$$\frac{25 \times 10^6}{640 \times 480} = 81\ fps$$

where 25×10^6 refers to the main clock frequency in the SoC implementation and 640×480 the total number of pixels. The throughput in Table 3.3 is computed by multiplying the Video Bus clock frequency and the bit width of the memory bus. In our design, we use 64 bits to store the 3-mixtures Gaussian model, and 8 bits for the input image. Thus the bit width of the memory bus is totally 72 bits, and the throughput is

$$\frac{25 \times 10^6 \times 72}{1024 \times 1024 \times 8} = 214\ Mega\ Bytes$$

Table 3.1. Design summary.

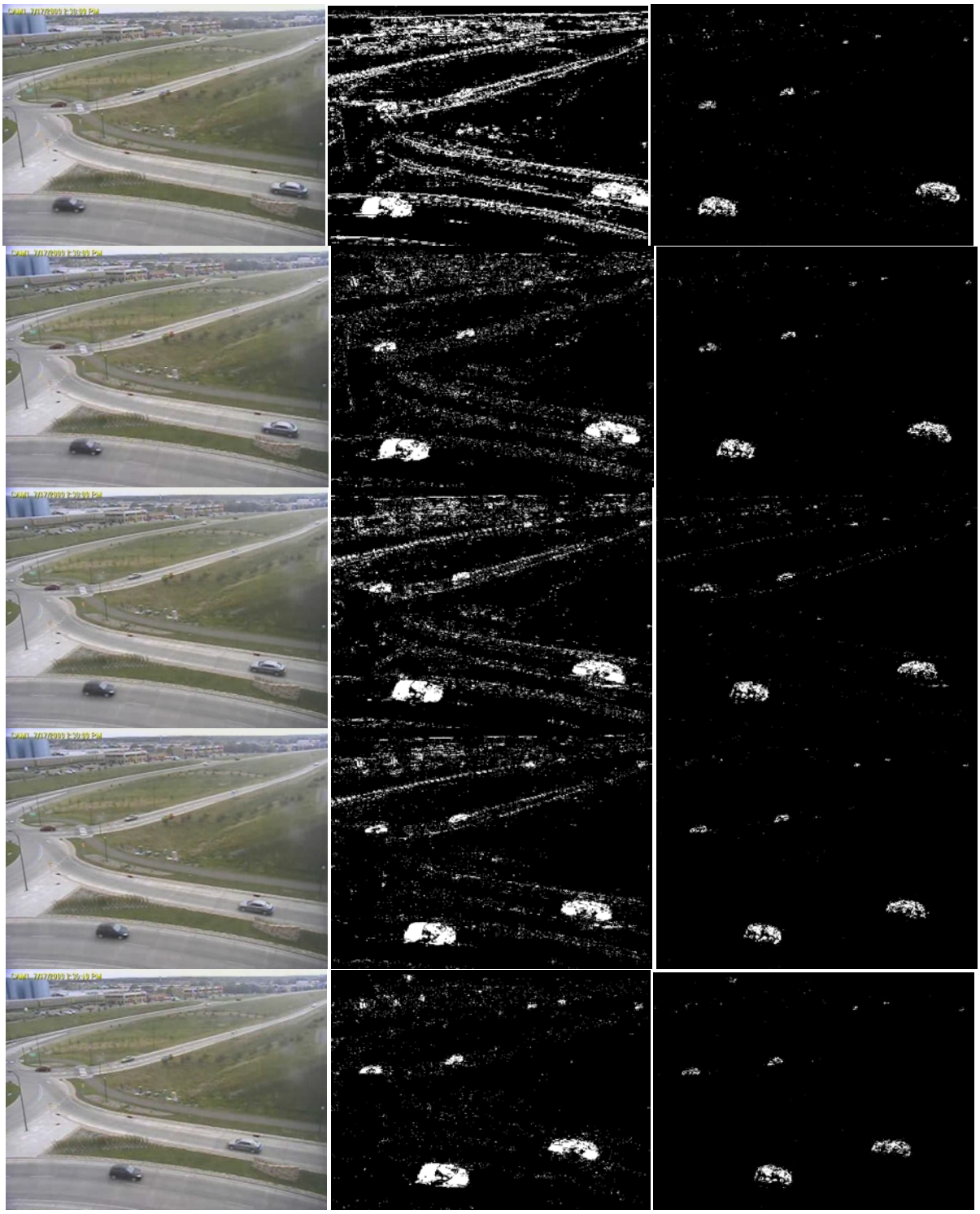
Logic	Number of used	Utilization
Slice Flip Flops	12,410	26%
4-input LUTs	15,921	33%
Occupied Slices	14,265	60%
DCMs	2	25%
BRAMs	77	61%
DSP48As	11	8%

Table 3.2. Hardware complexity of different blocks.

Logic Block	Flip Flops	4-input LUTs	BRAMs
Camera In	10	0	0
Camera Processing	992	1073	3
Gamma	95	62	3
MPMC	7040	7933	43
DVI out	29	1	0
UART	146	134	0
I^2C	382	494	0
MicroBlaze	1676	2639	4
MoG IP	1278	2481	0

Figure 3.8 shows the segmentation results for a pre-recorded video of a practical traffic scene, a roundabout located in Cottage Grove, Washington County, Minnesota. For comparison purpose, the segmentation results from the hardware implementation are shown together with those from the original software implementation for 10 consecutive image frames. It can be seen that they are very comparable. In general, the segmentation results from the software implementation has less noise due to the original algorithm and some additional processing steps while the hardware implementation runs the modified algorithm with a small sacrifice of quality. In fact, it is noted that the segmented vehicles from the hardware approach are more solid than those from the software approach, and this is due to additional processing steps performed in the software implementation other than the original MoG algorithm.

It is also important to note that camera shaking is not a major concern anymore for the MoG algorithm compared to traditional background subtraction algorithms. All the 10 images encountered constant camera shaking, but the segmentation results in most cases do not suffer much from that (except that first one) due to the multi-mode modeling capability of the MoG algorithm discussed in Chapter 2.2.



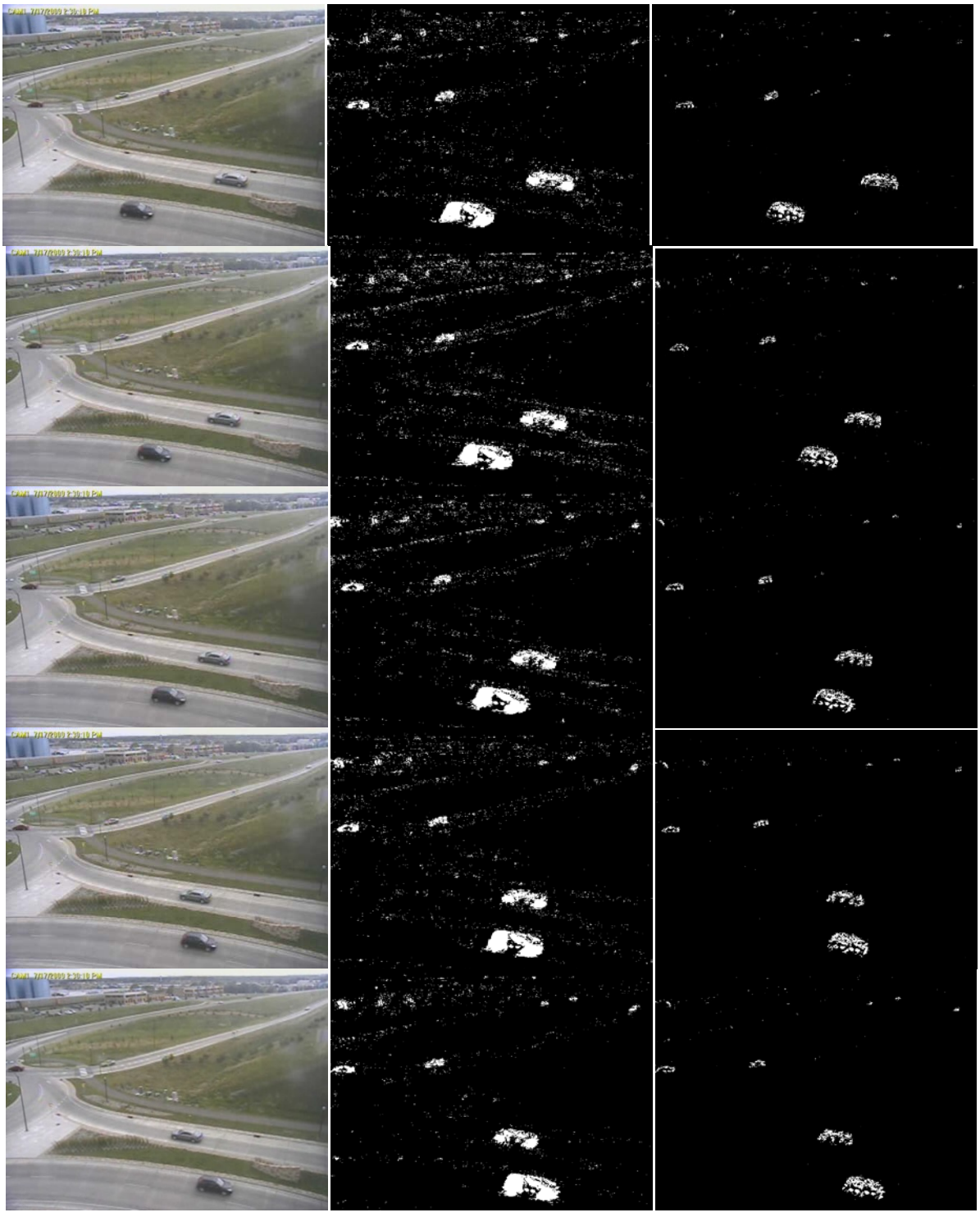


Figure 3.8. Segmentation results for a practical traffic scene (α= β = 0.0625). left: the original image, middle: the segmentation results from the hardware implementation, right: the segmentation results from the software implementation.

Another sample output of the video segmentation system in an office setting is shown in Figure 3.9. It can be seen that the moving object, the person in this case, is clearly identifiable in spite of incomplete detection in the sense that the body part is not segmented as foreground.

Finally, we made a comparison of the proposed design with the one proposed by Jiang et al [21]. The results are shown in Table 3.4. Note that the theoretically maximum frame rate of our design is 81fps for the image resolution 640×480, which is much higher than the one proposed by Jiang et al [21]. This is mainly because our hardware design uses totally 21-stage pipeline, which has much shorter critical path than the one proposed by Jiang et al [21]. Basically, the two systems have almost the same performance. Our system consumes more hardware resources than [21]. However, our system is more flexible than that in [21] at the expense of more hardware resources. For further development work such as video surveillance, object detection, and object tracking, our system is a more flexible platform due to the SoC architecture.

Figure 3.9. Segmentation result ($\alpha = \beta = 0.0625$).

Table 3.3. System parameters.

Resolution	640 ×480
Throughput	214 MB/s
Frame rate	30 fps
No. of Gaussians	3
PLB clock frequency	62.5 MHz
Video bus clock frequency	25 MHz
DDR2 clock frequency	125 MHz

Table 3.4. System comparison with [21].

	Jiang et. al [21]	The proposed design
No. of Slices	6,107	14,625
No. of Flip Flops	4,273	12,410
No. DCMs	5	2
No. of BRAMs	84	77
Resolution	640 × 480	640 × 480
Frame rate	25 fps	30 fps
Throughput	170 MB/s	214 MB/s
No. of Gaussians	3	3

3.5 A prototype vehicle detection system

So far in this Chapter, we have presented in detail how to implement the MoG algorithm based vehicle segmentation system in hardware. A complete vehicle detection system would incorporate several major components, including the camera-based imaging module, a vehicle segmentation system, a vehicle identification module and finally a message transfer module. In general, the vehicle segmentation system is the core of the overall vehicle detection system.

Finally, we show a developed prototype vehicle detection system with the MoG based vehicle segmentation system as the core. Figure 3.10 shows the complete system in a laboratory setting. The camera is used to capture images in real-time, which are then sent to the FPGA hardware which implements both the MoG-based vehicle segmentation system and the vehicle identification module. Note that vehicle identification itself is an important task as well and there are many algorithms proposed for object identification, such as the connected-component algorithm [1]. However, in our prototype system, we used the simple foreground-pixel counting based algorithm. This algorithm works as follows. First, given the camera view, a certain region is pre-selected and such a location could be places which need to be monitored, for instance, a region around the vehicle stops in an intersection. Second, for each image frame, the number of foreground pixels in the selected region is simply counted. If there are enough foreground pixels, then it is declared that a vehicle is present and in this case the detection result or signal is true. Otherwise no vehicle is detected. Finally, the detection result is sent through wired or wireless methods to a nearby message sign to alert drivers. In Figure 3.10, a wireless transmitter and receiver are used to send and receive the signal. The shown message sign is a solar-powered portable one with flashing lights and the warning message can be customized [27].

In the laboratory setting, we tested the prototype system. We played a pre-recorded video in one computer and let the camera focus on the screen and capture images in real-time (30fps). Then, the detection results were shown in another computer monitor (we used the computer monitor to emulate a practical message sign in the lab). The prototype system was validated with a few testing.

The compete vehicle detection system has been applied to facilitate vehicle merging in the laboratory. Figure 3.11 (one top view and one side view) shows the picture of two cases of two merging ramps (at Route 53 and I-35 in Duluth MN). The two merging ramps are shown in red and they eventually merge to one lane road. However, due to concrete walls between the two

ramps, drivers on one ramp are usually not aware of drivers on the other ramp, which are likely to cause and actually have caused accidents according to District I office of Mn/DOT, despite a yielding sign at one ramp. One application of the prototype vehicle detection system would then be to monitor the two ramps (for example as shown in the bottom figure of Figure 3.11) and issue warning if there are merging vehicles from both ramps. To test the system in the laboratory, again, a pre-recorded video of the merging ramps were played in one computer and the prototype vehicle detection system computed the detection results which were transmitted to be displayed in another computer. The prototype system had shown to be working in most cases, except when there are extremely large camera displacements. Note that one most important performance parameter in this case is the response time (the time delay between image capture and detection result display), which is preferred to be as small as possible in that a large delay would carry more risk of vehicle collision from vehicle merging. Our frame rate of the vehicle detection system is currently at 30fps and it was estimated that the response time is about 0.033 second. Within such a time delay, the vehicle would have traveled by less than 0.44 meter for a vehicle speed of 30 miles per hour (mph) or less than 0.6 meter of 40mph. We believe that such figures give some safety margins for vehicle merging.

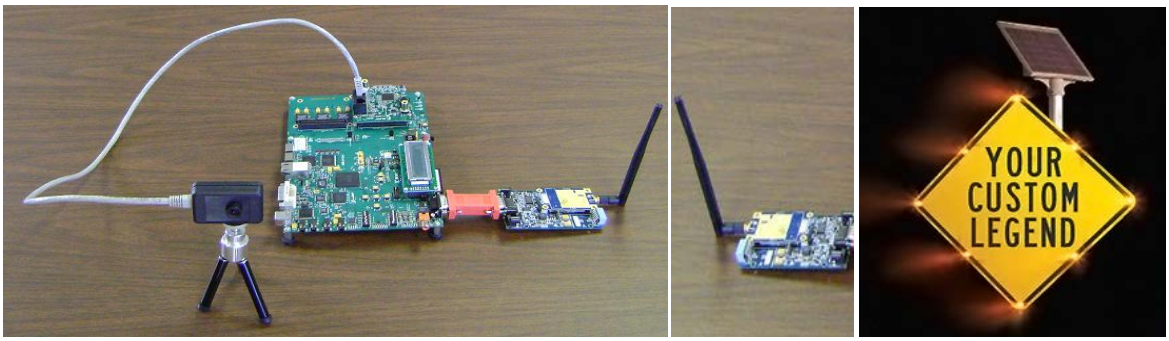


Figure 3.10. The prototype of the vehicle detection system.

3.6 Summary

In this Chapter, we first discuss how we modify the original MoG algorithm to allow efficient hardware implementation. Then, we present the implementation of a MoG IP for the modified MoG algorithm and the detailed circuit implementation for each processing step. We subsequently show the SoC architecture for implementation of the overall MoG-based vehicle segmentation system. Experiments of a lab scene and practical traffic scenes on the SoC architecture were performed and the results were satisfying. Finally, on the basis of the MoG-based vehicle segmentation system, we further developed a prototype real-time vehicle detection system by incorporating a vehicle identification module based on a simple foreground-pixel counting based algorithm and signal transfer interface based on a set of wireless transmitters and receivers. Such a prototype real-time vehicle detection system has been applied to facilitate vehicle merging for a practical traffic scene in the laboratory environments.

Google Maps - ©2012 Google

Figure 3.11. Top view and side view of the traffic scene where the vehicle detection system is applied to facilitate vehicle merging (the camera in the side view of the image shows the hypothesized location of installation).

4 Summary, Conclusion and Discussion

In this project, our main research task is to come up with a hardware implementation of the MoG algorithm and the corresponding SoC architecture for the MoG-based vehicle segmentation system to be applied in real-time vehicle detection systems. We first gave an introduction in Chapter 1 on the background of the project and why we had switched the vehicle segmentation algorithm from motion estimation in Phase I to MoG in Phase II. The capability of the MoG algorithm to cope with repeated camera shaking is a major advantage in that practical cameras always suffer from shaking due to wind. In Chapter 2 of the report, we explained in detail how the MoG algorithm segments vehicles and showed that it can help address camera shaking as validated with experimental results. In Chapter 3, we presented in detail the hardware implementation of the MoG algorithm and the overall SoC architecture for MoG-based vehicle segmentation system that allows online reconfiguration of some algorithm parameters that could affect segmentation quality. We showed experiment results of the vehicle segmentation system and evaluated its performance. We also compared the proposed hardware implementation of the MoG-based vehicle segmentation system to a state-of-the-art design reported in recent literature and showed that our design is competitive.

The main benefit of the project is that a hardware-based vehicle segmentation system and a vehicle detection system can provide a significant speed advantage compared to the traditional software-based vehicle detection system using computers. Such an advantage is very important in a few ways. First, time critical applications like security warning, vehicle collision warning, etc., require very fast response time, and the proposed hardware-based vehicle detection system has a very small delay from image capture to detection results. Second, such a hardware-based vehicle detection system is the first component of a complete video-based vehicle tracking system. Due to the small delay in processing the image, the hardware-based vehicle tracking system would have a much higher frame rate and a higher frame rate implies more tracking accuracy which leads to more accurate traffic data collection.

Regarding the hardware implementation of the MoG-based vehicle segmentation system, the main contribution of the proposed design is that the SoC architecture makes the system more flexible to adapt to different environments and makes it easier to develop other video processing IPs for applications such as video surveillance, object detection, and object tracking. The proposed design can support VGA resolution (640×480) at 30fps in real-time under 25MHz clock frequency. The maximum frame rate is 81fps.

Future work is planned as follows:

- Improvement of the prototype vehicle detection system.
  The focus of this project is hardware implementation of the vehicle segmentation system, which is the core of the complete vehicle detection system. Currently, the vehicle identification module is implemented using a simple foreground-pixel counting based scheme and this module can be improved for automated detection instead of manual specification of an interested region in the image. Also, the system can be integrated with a real variable message sign.
- Development of a complete hardware-based vehicle tracking system.

We demonstrated a hardware-based prototype vehicle detection system in this project built upon a vehicle segmentation system. In many ITS applications, especially those that require video-based traffic data collection, a complete vehicle tracking system is needed and the vehicle detection system is part of it. If a hardware-based implementation for the complete vehicle tracking system is necessary, which will allow very high-frame-rate vehicle tracking and significantly improve the accuracy of vehicle tracking and hence accuracy of traffic data collection, then the vehicle tracking algorithm needs to be implemented in hardware as well. Depending on the complexity of the vehicle tracking algorithm, this may be a very time-consuming task.

- Exploration of the new computer powers with graphical processing units (GPU). Recent advances in GPU has made personal computers more powerful than before and GPU-based computing has become more and more popular, especially in scientific and engineering applications involving computation-intensive operations. If vehicle segmentation and in general vehicle tracking is implemented on computers with GPU, the performance could be much improved and real-time, high-frame-rate vehicle detection is then possible. Note that a computer-based implementation of the vehicle detection system is a software approach as we discussed in Chapter 1, and the software approach does provide superior flexibility. Therefore, in the future, such a possibility of using GPU-based computers for implementation of a vehicle detection system and tracking system may be explored.

References

1. H. Tang, *Development of a New Tracking System based on CMOS Vision Processor Hardware: Phase I*, Final report, CTS 09-04, 2009, Center for Transportation Studies, University of Minnesota, Minneapolis, MN.
2. C. Stauffer, W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, June 1999, Ft. Collins, CO, pp. 246-252.
3. S. Chien, S. Ma, L. Chen, "Efficient moving object segmentation algorithm using background registration technique," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 7, pp. 577-586, 2002.
4. D. A. Migliore, M. Matteucci, M. Naccari, "View-based detection and analysis of periodic motion," *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, October 2006, Santa Barbara, CA, pp. 215-218.
5. J. Moon, D. Kim, and R. Park, "Video matting based on background estimation," *Proceedings of the World Academy of Science, Engineering and Technology*, vol. 2, January 2005, pp. 152-155.
6. Q. Zhang, R. Klette, "Robust background subtraction and maintenance," *Proceedings of the 17th International Conference on Pattern Recognition*, August 2004, Cambridge, U.K., pp. 90-93.
7. R. Cutler, L. Davis, "View-based detection and analysis of periodic motion," *Proceedings of the 14th International Conference on Pattern Recognition*, August 1998, Brisbane, Australia, pp. 495-500.
8. B. Lo, S. Velastin, "Automatic congestion detection system for underground platforms," *Proceedings of the International Symposium on Intelligent Multimedia, Video Speech Processing*, May 2001, Hong Kong, pp. 158-161.
9. F. Porikli, "Multiplicative background-foreground estimation under uncontrolled illumination using intrinsic images," *Proceedings of the IEEE Workshop Motion Video Computing* (WACV/MOTION'05), January 2005, Breckenridge, CO, pp. 20-25.
10. S. Haykin, *Adaptive Filter Theory*, 4th edition, 2001, Prentice-Hall, Upper Saddle River, NJ.
11. J. Zhong, S. Sclaroff, "Segmenting foreground objects from a dynamic textured background via a robust Kalman filter," *Proceedings of the International Conference on Computer Vision*, October 2003, Nice, France, pp. 44-50.
12. S. Messelodi, C. M. Modena, N. Segata, M. Zanin, "A Kalman filter based background updating algorithm robust to sharp illumination changes," *Proceedings of the 13th International Conference on Image Analysis Processing*, 2005, Cagliari, Italy, pp. 163-170.
13. C. Ridder, O. Munkelt, H. Kirchner, "Adaptive background estimation and foreground detection using Kalman-filtering," *Proceedings of the International Conference on Recent Advances in Mechatronics*, 1995, Istanbul, Turkey, pp. 193-199.

14. O. Javed, K. Shafique, M. Shah, "A hierarchical approach to robust background subtraction using color and gradient information," *Proceedings of the Workshop Motion Video Computing*, 2002, Orlando, FL, pp. 22-27.

15. L. Li, W. Huang, I. Y. H. Gu, Q. Tian, "Foreground object detection in changing background based on color co-occurrence statistics," *Proceedings of the 6th IEEE Workshop on Applications of Computer Vision*, Orlando, FL, 2002, pp. 269-274.

16. M. Harville, G. Gordon, J. Woodfill, "Adaptive video background modeling using color and depth," *Proceedings of the IEEE International Conference on Image Processing*, October 2001, Thessaloniki, Greece, pp. 90-93.

17. Y. Zhang, Z. Liang, Z. Hou, H. Wang, M. Tan, "An adaptive mixture Gaussian background model with online background reconstruction and adjustable foreground mergence time for motion segmentation," *Proceedings of the IEEE International Conference Industrial Technology*, December 2005, Hong Kong, pp. 23-27.

18. H. Wang, D. Suter, "A re-evaluation of mixture-of-Gaussian back-ground modeling," *Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing*, March 2005, Philadelphia, PA, pp. 1017-1020.

19. Y. Xu, D. Xu, Y. Liu, A. Wu, "Illumination invariant motion estimation for video segmentation," *Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing*, May 2004, Quebec, Canada, pp. 737-740.

20. A. Elgammal, D. Harwood, L. Davis, "Non-parametric model for background subtraction," *Proceedings of the European Conference on Computer Vision (ECCV)*, 2000, Dublin, Ireland, pp. 751-767.

21. H. Jiang, H. Ardo, V. Owall, "A hardware architecture for real-time video segmentation utilizing memory reduction techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 2, pp. 226-236, 2009.

22. "Spartan-3A DSP FPGA video starter kit user guide," 2008, Xilinx, Inc., San Jose, CA.

23. I. Pavlidis, V. Morellas, P. Tsiamyrtzis, S. Harp, "Urban Surveillance Systems: From the Laboratory to the Commercial World," *Proceedings of IEEE*, vol. 89, no. 10, pp. 1478-1497, 2001.

24. "Processor Local Bus (PLB) v4.6 (v1.04a) - Product Specification," 2009, Xilinx, Inc., San Jose, CA.

25. "Microblaze processor reference guide," 2009, Xilinx, Inc., San Jose, CA.

26. "Multi-port memory controller (MPMC) (v5.04.a) - Product Specification," 2009, Xilinx, Inc., San Jose, CA.

27. "Custom BlinkerSign," 2011, Traffic & Parking Control Co., Inc., Brown Deer, WI.