HARDWARE IMPLEMENTATION OF AN ACTIVE FEATURE TRACKER FOR SURVEILLANCE APPLICATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BERKAN SOLMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JULY 2008

Approval of the thesis:

HARDWARE IMPLEMENTATION OF AN ACTIVE FEATURE TRACKER FOR SURVEILLANCE APPLICATIONS

submitted by **BERKAN SOLMAZ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering Dept.**

Prof. Dr. Gözde Bozdağı Akar _____
Supervisor, **Electrical and Electronics Engineering Dept.**

Examining Committee Members:

Assoc. Prof. Dr. A. Aydın Alatan _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Gözde Bozdağı Akar _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. İlkay Ulusoy _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Çağatay Candan _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Yasemin Yardımcı Çetin _____
Informatics Institute, METU

Date: July 17, 2008

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Berkan SOLMAZ

Signature :

ABSTRACT

HARDWARE IMPLEMENTATION OF AN ACTIVE FEATURE TRACKER FOR SURVEILLANCE APPLICATIONS

Solmaz, Berkan
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Akar, Gözde Bozdağı

July 2008, 88 pages

The integration of image sensors and high performance processors into embedded systems enabled the development of intelligent vision systems. In this thesis, we developed an active autonomous system to be used for surveillance applications. The proposed system detects a single moving object in the field of view automatically and tracks it in a wide area by controlling the pan-tilt-zoom features of the camera. The system can also go to an alarm state to warn the user.

The processing unit of the system is a Texas Instruments DM642 Evaluation Module which is a low-cost high performance video & imaging development platform designed to develop and evaluate video based applications.

Keywords: Tracking, Background Modeling, Features, Digital Signal Processor, Pan-Tilt-Zoom Camera, Surveillance.

öz

GÖZETLEME UYGULAMALARI İÇİN BİR AKTİF ÖZNİTELİK İZLEYİCİNİN DONANIM GERÇEKLEŞTİRİMİ

Solmaz, Berkan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Temmuz 2008, 88 sayfa

Görsel algılayıcılar ve yüksek performanslı işlemcilerin bir araya getirilmesi akıllı görme sistemlerinin geliştirilmesine olanak sağlamıştır. Bu tezde, gözetleme uygulamalarında kullanmak üzere aktif, özerk bir sistem geliştirilmiştir. Önerilen sistem, kameranın bakış açısı içinde bulunan hareketli bir nesneyi otomatik olarak algılamakta ve nesneyi kameranın yatay-eğim-optik kaydırma özellikleriyle geniş bir alanda takip etmektedir. Ayrıca sistem bir alarm durumuna geçerek kullanıcıyı da uyarabilmektedir.

Sistemin işleme birimi düşük maliyetli, yüksek performanslı video ve görüntüleme uygulamaları geliştirme ve değerlendirme platformu olan Texas Instruments'a ait DM642 Değerlendirme Modülüdür.

Anahtar Sözcükler: İzleme, Arkaplan Modelleme, Öznitelik, Sayısal İşaret İşlemci, Yatay-Eğim-Optik Kaydırmalı Kamera, Gözetleme.

To my family

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Prof. Dr. Gözde Bozdağı Akar for her guidance, encouragement, support and patience throughout the preparation of this thesis. I am grateful to Assist. Prof. Dr. İlkay Ulusoy for her help and support during this thesis. Thanks to Anıl Aksay, Yusuf Bediz and Ahmet Oğuz Öztürk for their encouraging me in the development, for their valuable suggestions and support.

Thanks to Murat Deniz Aykın, Emrah Bala, Mehmet Oğuz Bici, Özlem Pasin and all colleagues from Multi Media Research Group, for their suggestions and availability to help me whenever it was necessary and also for the pleasant working environment.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

DSP	Digital Signal Processor
BG	Background
FG	Foreground
SGM	Single Gaussian Model
CCD	Charge Coupled Device
EM	Expectation Maximization
MGM	Mixture of Gaussians Model
FPS	Frame per Second
KLT	Kanade-Lucas-Tomasi
ROI	Region of Interest
PETS	Performance evaluation of tracking and surveillance
PTZ	Pan-Tilt-Zoom
CPU	Central Processing Unit
RAM	Random Access Memory
SRAM	Static Random Access Memory
L1	Level 1
L2	Level 2
DMA	Direct Memory Access
RISC	Reduced Instruction Set Computer
EDMA	Enhanced Direct Memory Access
Y/C	Luminance and Chrominance
I/O	Input/Output
VLIW	Very Long Instruction Word
ILP	Instruction Level Parallelism
TI	Texas Instruments
EVM	Evaluation Module

LED	Light Emitting Diode
GPIO	General Purpose Input Output
PCI	Peripheral Component Interconnect
JTAG	Joint Test Action Group
McBSP	Multi-Channel Buffered Serial Port
VISCA	Video System Control Architecture
CCS	Code Composer Studio
PAL	Phase Alternating Line
GEL	General Extension Language
CSL	Chip Support Library

CHAPTER 1

INTRODUCTION

1.1 Background

Surveillance involves monitoring the movements of people or objects. In past, a complete surveillance system consisted of a camera, a monitor and a video recorder. This system did not have the ability to process the acquired signals or analyze the scene. In more advanced surveillance and vision systems, the cameras operate as simple sensors and the captured video is processed in a central processing unit which is generally a personal computer. However, the whole video stream needs to be transmitted to the processing unit by a high-cost and distance limited connection. Therefore, it is more feasible to integrate the processing unit and the camera within a stand-alone system and to run the video/image processing algorithms on this system. Integration of image sensors and high performance processors into embedded systems enabled the development of capable stand-alone systems.

The active feature tracker that is presented in this thesis is an intelligent vision system that performs not only video capturing but also processing and extracting information from video stream without the need for an external processing unit. It can also interface with other devices to report the results and events to the users.

The presented system has all of the essential components for video/image processing and also some additional components. These features can are:

-Pan-tilt-zoom camera for capturing analog video
-Video decoder for digitizing analog video
-Digital signal processor (DM642)
-Code Memory and Data memory (Ram, Flash)
-Communication interface (RS-232)
-General Purpose I/O lines
-Illumination devices (LEDs)
-Video encoder for video output

The implemented system is appropriate for many tasks. As well as being used for active feature tracking for surveillance in this thesis, the hardware of this system can be programmed to execute various types of image/video processing routines. Therefore conventional PC based or central processing unit based systems will be unnecessary. This system can also compete with personal computers in terms of functionalities. The compactness and stand-alone operation of this type of vision systems make integration and mounting easier. This also results in a decrease in costs, for instance no switching cabinets are required for the cameras. This type of systems is more reliable and can work for long time without being restarted. Nevertheless, this type of systems do not have sophisticated user interface like PC based systems.

The presented stand-alone system, including a high performance digital signal processor, can be a perfect choice for applications of distributed vision and for applications where multiple independent and asynchronous cameras are required. For instance, many of these systems can be distributed along a production line or at multiple inspection or surveillance points.

In recent years, due to the increasing demand for intelligent surveillance systems, there are various applications being developed. Haritaoglu [1] et al. proposed a surveillance system that identifies

shopping groups by detecting and tracking people while they wait in a checkout line or service counter. Hampapur et al. [2] developed a system to perform 3D head detection in a room by using multiple cameras.

Although the presented system has many advantages, performing high-level image analysis algorithms on embedded systems requires high processing power and it is the most critical task for the development of embedded vision systems.

1.2 Problem Definition and Motivation

Video surveillance systems of today generally consist of closed-circuit TV systems. These systems accomplish a loopback of images to a monitor for people who try to examine events. However, the people viewing the images on monitors cannot be so consistent and may lose their concentration in time. With the increasing technology, video surveillance systems started evolving from traditional closed-circuit TV systems to intelligent stand-alone security systems which can operate more consistently and less dependent on people. High processing power of the new digital signal processors enabled more functional systems to be developed. These systems have many more features when compared to the traditional systems which can only capture video signals and record them. They automatically analyze the video stream and perform actions such as warning the security guards in case of an important event.

This thesis describes the hardware implementation of an intelligent system with pan-tilt-zoom features for surveillance applications. The hardware of the system consists of a Texas Instruments DM642 Digital Video Processing board and a pan-tilt-zoom camera. The presented system detects a single moving object in the field of view automatically and tracks it in a wide area by controlling the pan-tilt-zoom features of

the camera. The system can also go to an alarm state to warn the user if necessary.

1.3 Organization of the Thesis

The organization of the thesis is as follows:

In Chapter 2, the theoretical bases of background modeling, feature extraction and tracking, which are necessary for moving object detection, are given. Major approaches are discussed by presenting their strengths and weaknesses.

Chapter 3 illustrates the hardware of the presented system by stating the most critical parts of the hardware and points out their general specifications.

In Chapter 4, the whole system implementation and integration are described. The hardware used for the development and the developed software are explained in details. The testing and verification steps are also addressed.

In Chapter 5, the setup of experiments for the system and the results obtained from the execution of the developed software on PC and on DSP using sample or real-time video sequences are given.

Finally, Chapter 6 provides the conclusions for the overall study with mentioning open points for possible future studies.

CHAPTER 2

MOVING OBJECT DETECTION

2.1 Background Modeling

Background modeling is a fundamental task for many vision systems. The aim is to detect the background scene in a suitable way, detect the changes in the scene and separate the background and foreground objects carefully. The more accurate the background model, the more accurate the detection of foreground objects. The most critical part in background modeling is to build an adaptive model for the background, as in most scenes where the background shows a varying behavior in time and space. Especially natural scenes result in many difficulties on background modeling since they are usually dynamic and including illumination changes that may result from cloud cover and as well as variations that result from moving background objects such as tree leaves, rain, snow and sea waves. A robust background modeling algorithm should also deal with cases when new objects are introduced into the background or when a background object is removed from the scene. Furthermore variation due to blocking of the light source in indoor videos or the shadows of the moving objects can cause problems. Even in a static scene, frame to frame changes may occur due to noise and uncertain camera movements. An efficient background model needs to overcome these problems.

In security applications, many objectives such as object tracking and action analysis are dependent on background modeling. Therefore, it is the first task of the proposed system in the thesis. There are various

background models that have been introduced with various characteristics. Popular approaches will be explained in following sections.

2.1.1 Background Subtraction and Temporal Smoothing

Background subtraction is a basic and common method for discriminating a moving object from the background scene. The captured frames are subtracted from the estimated background image and the result is thresholded to generate the objects of interest.

Most of the real-time security systems use this method in order to detect the regions of the image that have changed. The simplest way of acquiring the background image is to set a prior frame as the reference frame when there is no foreground object in the scene. Then using pixel-by-pixel frame difference between the reference frame and the captured frames allows us to find the foreground objects in the captured frames. Thresholding the difference results in a binary foreground image that consists of two gray levels; 0 (black) indicates the background pixels and 1 (white) indicates the foreground pixels. The foreground image is given by:

$$F(x,y,t) = \begin{cases} 1, & |I(x,y,t) - B(x,y)| > Thr \\ 0, & |I(x,y,t) - B(x,y)| < Thr \end{cases} \quad (2.1)$$

F(x,y,t) and I(x,y,t) represent the intensity vectors for the pixels of foreground image and captured frame at (x,y) location at time t respectively. B(x,y) is the intensity vector for the pixel at location (x,y) of background image and Thr is the threshold value.

This method is useful when the background is totally static. However, it has so many errors when the background is not relatively static. For

instance, when the illumination level changes depending on the weather, position of clouds and sun or when the sun light lessens, it affects the performance. Another problem of this method is that the present foreground objects while setting the reference frame remain as permanent background objects. Moreover slight camera movements also result in errors in detecting the background. Leaves of a tree when there is wind results in errors. A flickering monitor or TV screen is also a problem for this method.

An improved method for determining background image, which is known as Temporal Smoothing, is accomplished by averaging the captured frames over time in order to have a better estimate. The background estimate can be found by:

$$B(x,y,t)=\frac{1}{t}\sum_{t'=1}^{t} I(x,y,t') \tag{2.2}$$

where I(x,y,t) is the intensity vector for the pixel at (x,y) location at time t and B(x,y,t) is actually the mean color of the pixel at (x,y) at time t. In order to reduce the required memory and the required number of multiplications and additions, the formula can be computed incrementally as:

$$B(x,y,t)=\frac{t-1}{t}B(x,y,t-1)+\frac{1}{t}I(x,y,t) \tag{2.3}$$

Temporal smoothing performs well in scenes where objects move continuously and the background is visible for a significant portion of the time. However it fails when the foreground objects move slowly since those objects will be assigned significantly into the averaged background. In addition, when the scene is occupied by many foreground objects, the background becomes less visible. Therefore, this method has errors forgetting the background image. This

technique has a particularly long recovery time for the background when a new static object is inserted to the scene or when a background object is removed from the scene. A change in the illumination level over time is another problem of this method. This problem can be solved by using a moving window average of frames or using an exponential forgetting. The weights of past frames decrease exponentially and their effect on the background image reduces. Using this approach, the background image can be obtained by:

$$B(x,y,t) = (1-a) * B(x,y,t-1) + a * I(x,y,t) \tag{2.4}$$

where 1/α is the forgetting factor and α is the learning rate. In this way the memory requirements are also reduced. In simple average case, the memory requirement is frame size times the number of all frames, whereas in moving average case the required memory is the just the frame size. This approach also cannot adapt to the scenes with tree leaves, snow, rain or slowly moving objects.

Temporal smoothing may be performed in various ways. For example, Karmann and von Brandt [3] use a Kalman filter and the Wallflower algorithm [4] uses a Wiener filter instead of exponential smoothing.

Heikkila and Olli [5] applied two corrections to the method:
-If a pixel is observed in the foreground for more than a number of frames in a specified number of last frames, then it is modified to be in the background by setting:

$$B(x,y,t) = I(x,y,t) \tag{2.5}$$

With this modification the method can adapt to the sudden illumination changes and to the insertion of new static objects.

-If the intensity of a pixel changes frequently, it is masked out from being in the foreground. This modification is helpful in conditions when there are object that cause fluctuations in the illumination level of environment such as flickering monitors, tree leaves in wind or sea waves.

Moving average approach can also be modified in order not to take into account the foreground pixels while estimating the background by performing:

$$B(x,y,t+1)=\begin{cases} a*I(x,y,t)+(1-a)*B(x,y,t), & if \quad I(x,y,t) \quad is \quad in \quad BG \\ B(x,y,t), & if \quad I(x,y,t) \quad is \quad in \quad FG \end{cases} \quad (2.6)$$

After background subtraction, morphological operations are applied to the foreground. Generally, a 3x3 mask is used for dilation and erosion operations. Then, the foreground object is segmented by connected component analysis. The same procedure is also applied in the methods Single Gaussian Distribution Model (Section 2.1.2).

While background subtraction and temporal smoothing methods are useful in well-defined and short-time tracking applications without significant changes in the scene, they may have many errors that accumulate over time in more complex scenes. These methods cannot cope with complex backgrounds, and have a single, predetermined threshold for the entire scene.

2.1.2 Single Gaussian Distribution Method

As mentioned in the previous sections, the background scene is generally not static and changes over time due to variation in illumination levels. Therefore the background model needs to be estimated and updated properly in time by analyzing the frame sequence. Pixels of the background model can be represented by

intensity vectors that consist of numerical values for all color channels. However, the most popular approaches for background modeling are based on probabilistic models.

In many methods, pixels of background models are represented by distributions in order to express the variable behavior of the background. The goal of these sophisticated and adaptive methods is to estimate the probability density function of every pixel of the frames and to compute the probabilities for every pixel to be a part of the background scene. These probability density functions can be estimated in many ways.

The most popular and a simple method is to describe the intensity values of each pixel in a video sequence with a single Gaussian distribution. Therefore the mean and variance of Gaussian distributions are updated over time. Many background modeling algorithms rely on this method. The decision of a pixel to be in background or in foreground is done by analyzing the deviation of the intensity of the pixel from the estimated mean value. This is the most important advantage of these methods when compared to the basic methods where the thresholds are constant. The sensitivity of these methods is dependent on the pixel that is analyzed. After thresholding operation, a binary image is obtained as in the previous methods.

In Pfinder system of Wren et al.'s [6], which is a good example of single distribution background models, the background scene is represented by using a single Gaussian distribution per pixel. The distributions are initialized by estimating the mean and variance parameters of the background pixels independently when there are no foreground objects in the scene. Then these parameters of the background distributions are updated using a simple adaptive filter by:

$$\overline{\mu}_t = a * \overline{I}_t + (1-a) * \overline{\mu}_{t-1} \tag{2.7}$$

$$\overline{\sigma}_t^2 = a * (\overline{I}_t - \overline{\mu}_t)(\overline{I}_t - \overline{\mu}_t)^T + (1-a) * \overline{\sigma}_{t-1}^2 \tag{2.8}$$

where α is the learning rate and I_t is the captured frame at time t. I_t can be represented by the vector of intensities $[\Phi_Y, \Phi_U, \Phi_V]^T$ in YUV color space. Then the log likelihood is quantified in order to specify the foreground and background pixels as:

$$\ell_t = -\frac{1}{2}(\overline{I}_t - \overline{\mu}_t)^T (\overline{\sigma}_t^2)^{-1} (\overline{I}_t - \overline{\mu}_t) - \frac{1}{2}\ln | \overline{\sigma}_t^2 | - \frac{m}{2}\ln(2\pi) \tag{2.9}$$

A pixel is identified as in the foreground if the log likelihood is smaller than a specified value. The last step is to find the connected components as it is in many methods. For grayscale images, rather than using the log likelihood function, a pixel is specified as foreground or background by testing with a threshold value which is taken as a constant times the variance of that pixel.

$$|I - \mu| > Threshold \tag{2.10}$$

This system is reported to work effectively in indoor scenes where there are not many variations in the background, but it cannot cope with multimodal backgrounds. While the intensity values of pixels in a background with slow changes can be properly described by recursively updating the parameters of these single distribution models, these models will have difficulties in estimating complex and varying background scenes. When there are actions such as change in illumination levels or slight movement and variation of background objects in the scene, one distribution will not be enough to represent the behavior of a single pixel over time. Therefore multimodal

background models, namely mixture of distributions, will need to be used in case of moving backgrounds.

2.1.3 Mixture of Gaussians Method

Stauffer and Grimson [7] proposed a method which suggests that the samples of a pixel observed over time in many scenes can be described by more than one process. Therefore, the intensity of each pixel is modeled by a mixture of weighted Gaussian distributions instead of modeling with one particular type of distribution which is usually a single Gaussian distribution. The weight of a Gaussian distribution is dependent on the permanence of that distribution to be the dominant color in the scene. This method makes it possible to handle natural backgrounds which are generally multimodal and not static.

While the simpler methods and Single Gaussian Distribution Model that are previously discussed are more appropriate for particular tasks and for modeling simpler background scenes, Mixture of Gaussians method has better performance for modeling more complex and time-varying background scenes and works efficiently in many cases. This successful background model can cope with daily variations in lightning, repetitive motions in the scene, and also insertion of objects to the scene or removal of objects from the scene. The model avoids the slow moving foreground objects being identified as background objects since the moving objects have a larger color variance than the background. The method also adapts quickly in case of appearing ghosts or fast changing illumination.

The proposed system in this thesis uses Mixture of Gaussians modeling method with deterministic thresholding for background modeling. In this method, every pixel in the scene is modeled by a mixture of Gaussian distributions. The two main tasks that this method deals with are updating the weights and parameters of each Gaussian distribution

with respect to the captured frames and specifying the Gaussian distributions that represent the background. A pixel is considered as in the foreground if its intensity does not fit any of the Gaussians which represent the background. This method has superior performance and adapts to changes in conditions if the parameters are set in a suitable way.

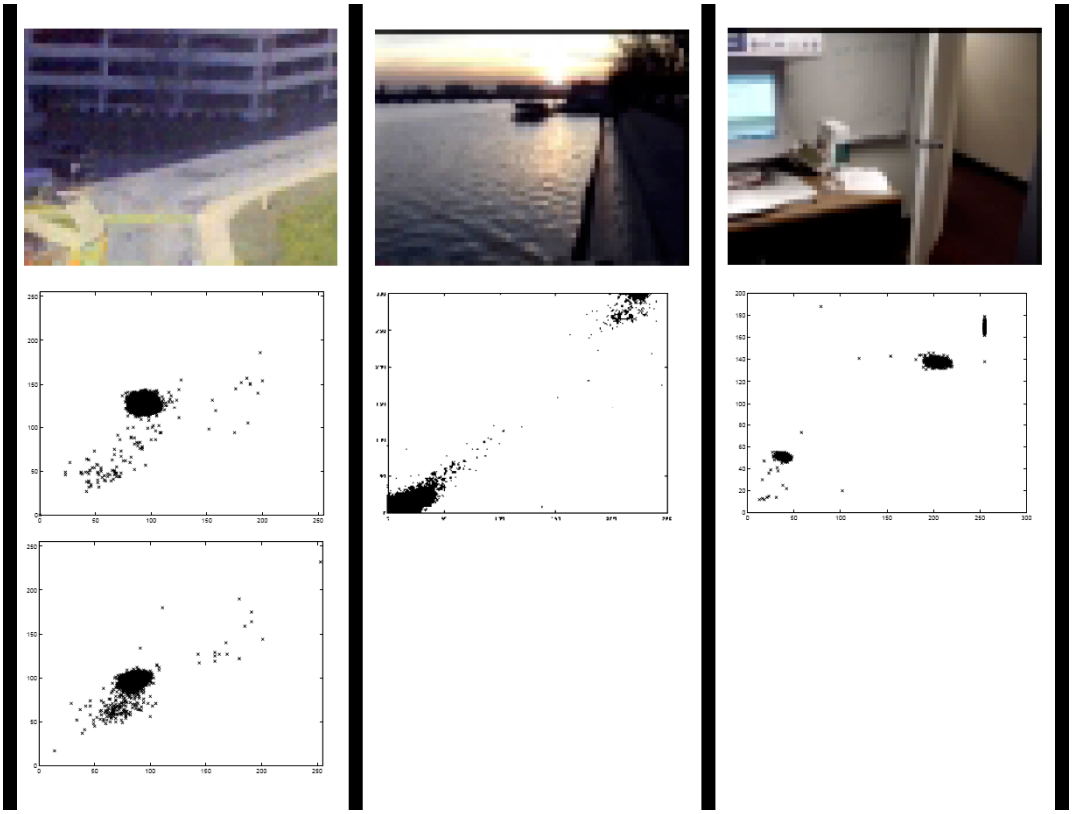
The learning constant *a* and the proportion of data that need to be used for describing background *T* are the two important parameters to be specified in this method.

For each captured frame, the parameters of the Gaussian distributions are updated, and the Gaussian distributions are evaluated to identify which ones most likely represent the background scene. If the pixel values do not match any of the background distributions they are identified as foreground pixels and these are grouped using connected components. Finally, the connected components will be tracked from frame to frame using a tracking algorithm.

The values of a pixel over time are called a pixel process. A pixel process consists of scalars for grayscale images and vectors for color images. The history of a pixel is given by all its values up to a time t as:

$$\{X_1, X_2, \ldots, X_t\} = \{I(x_0, y_0, i) \quad for \quad 1 \le i \le t\} \qquad (2.11)$$

where $I(x_0,y_0,i)$ represents the pixel value of the captured image sequence at location (x_0, y_0) and at time i. Some pixel processes that are taken by Grimson et al.'s work [7] are illustrated in Figure 2.1 using (R, G) scatter plots.



(a) (b) (c)

Figure 2.1: Scatter plots of pixels obtained on the images over time [7]

In the first row of Figure 2.1, one frame per each of the used scenes are illustrated. In the other rows there are the scatter plots. For (a) two scatter plots of the same pixel in different time periods are given. If we observe the plot we can see that the values of the pixel change in time. Such a scenario may happen due to shadowing effect. In such a case, a system with a single predetermined threshold would fail. In case (b) the pixel has a bi-modal character which means that it has two different values over time. This is caused by the variation of color on the surface of water. In case (c), another pixel with bi-modal character is analyzed which results from the flickering of monitor in the scene. It is observed in the (R, G) scatter plots that an adaptive system with deterministic thresholding which can handle multi-modal representations is desired for these cases.

With a totally static background and static lightning, the pixels have constant values. Under the assumption of independent Gaussian noise in the pixel values, the pixel processes can be described by single Gaussian distributions. However the scenes generally have variations in illumination and moving objects, so the Gaussian should adapt to those changes. In case of an inserted static object in the background, the corresponding pixels will be identified as in the foreground for a long time period which will result in accumulated errors in background modeling. Therefore, recent observations need to have higher significance in estimating the parameters of Gaussians. Moving objects in the scene also cause variations. Even if the moving object has a consistent color, it causes a variance more than a static object in the scene.

The recent history of each pixel, $\{X_1, X_2, ..., X_t\}$, is modeled by a mixture of Gaussian distributions, where X_t is the current pixel value vector that consists of red, green, blue component intensities.

$$X_t = (x_t^r, x_t^g, x_t^b) \tag{2.12}$$

The probability of observing the current pixel value is

$$P(X_t) = \sum_{i=1}^{K} w_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{2.13}$$

where K is the number of Gaussian distributions in the mixture which is specified according to the available memory and computational power of the system, $w_{i,t}$ is an estimate of the weight of the i^{th} Gaussian in the mixture at time t, $\mu_{i,t}$ and $\Sigma_{i,t}$ are respectively the mean value and the covariance matrix of the i^{th} Gaussian in the mixture at time t, and η is a Gaussian probability density function. η, $\mu_{i,t}$ and $\Sigma_{i,t}$ are given respectively by:

$$\eta(X_t,\mu,\Sigma)=\frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}e^{-\frac{1}{2}(X_t-\mu_t)^T\Sigma^{-1}(X_t-\mu_t)} \tag{2.14}$$

$$\mu_{i,t}=(\mu_{i,t}^r,\mu_{i,t}^g,\mu_{i,t}^b) \tag{2.15}$$

$$\Sigma_{i,t}=\begin{pmatrix}\sigma_r^2 & 0 & 0\\ 0 & \sigma_g^2 & 0\\ 0 & 0 & \sigma_b^2\end{pmatrix} \tag{2.16}$$

Under the assumption that the color components are independent with equal inner variances, the covariance matrix is assumed to be in the diagonal matrix form for computational simplicity as in Eq. 2.17. This assumption allows us to avoid a costly matrix inversion at the expense of some accuracy.

$$\Sigma_{i,t}=\sigma_i^2 I \tag{2.17}$$

The distribution of a pixel value is modeled by a mixture of Gaussian distributions. The most recent value of a pixel is described by one of the Gaussians in the mixture and is used for updating the model. Considering the pixel process to be stationary, expectation maximization (EM) algorithm can be used for maximizing the likelihood of observed data. However, pixel processes vary in time in natural scenes. Therefore, an approximate method which uses only the newest sample and integrates the new data by standard learning rules is used.

Instead of implementing an EM on a window of recent data for each pixel of a frame, a computationally simpler online K-means approximation is implemented in the method. Each pixel value, X_t is tried to be matched to one of K Gaussian distributions. A matching to a Gaussian is done if the pixel value is within a distance of 2.5 standard deviations from the mean of that distribution. This per pixel per

distribution threshold is very efficient when the different regions have different lighting. Analyzing the case (a) of Figure 2.1, it is observed that the noise in shaded regions is less significant than the noise in lighted regions. Using a uniform threshold often results in objects disappearing when they enter shaded regions.

In case of no matches between a new pixel value and the existing distributions, the distribution with the smallest likelihood with respect to the new pixel value is discarded and a new distribution with a mean set as this new pixel value, with a variance set as a high predetermined constant, and with a small weight is created instead of the discarded one. Therefore, random short term variations in the scene do not form a dominant distribution in the model.

The update for the weights of distributions is done by:

$$w_{i,t} = (1-a)w_{i,t-1} + aM_{i,t} \tag{2.18}$$

In the formula, α is the learning rate typically taken between 0.3 and 0.7 and 1/α specifies the speed of change of the parameters of distribution. $M_{i,t}$ is a parameter that is taken 1 for the matching distribution and 0 for the others. Thus the weight of the matching distribution increases whereas the weights of other distributions decrease. After all weights are updated, they are renormalized. $w_{i,t}$ *i*s actually a causal low-pass filtered average of the posterior probability that pixel values have matched the i^{th} model given observations from time 1 to *t*.

The update procedure is different for matching and non matching distributions. With each captured frame, the mean value μ and the variance σ^2 parameters of only the distribution that matches the new

pixel value are updated by using a same type of causal low-pass filter except the use of matching observation X_t in the estimation:

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t \quad (2.19)$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \quad (2.20)$$

$$\rho = a\eta(X_t \mid \mu_k, \sigma_k) \quad (2.21)$$

The mean value μ and the variance σ^2 parameters for unmatched distributions are not updated. In this method, when a color becomes a part of the background, the other distributions are not totally discarded. The background color remains in the mixture until it has the K^{th} most likelihood of being the dominant color and a new color is observed. Thus, when an object, which is stationary for a time period to be a part of the background, moves, the distribution of the original background scene still exists in the mixture with same mean and variance parameters, but with a smaller weight. So, the original distribution can quickly re-incorporate into the background.

The parameters of the mixture model change as the background processes change. The Gaussian distributions which have the most supporting evidence in time and the least variance are most likely the background processes. For instance, a static and consistent object in the scene causes higher supporting evidence which can be taken as the weight of that distribution and a lower variance than an object which occludes the background object. The occluding object does not match any distribution at first and results in a new distribution to be created or an existing one to have a larger variance. The variances of the moving objects are generally larger than the background scene while they keep on moving.

After estimating the parameters of the mixture, a good idea to decide on which processes model the background more effectively is to sort

the distributions in a list according to the fitness value ω/σ which increases with the more supporting evidence of a distribution and a lower variance. So the most likely background distributions remain on top of the list while the less probable or transient background distributions move towards the bottom of the list and are replaced by new distributions. Then the first B distributions of the sorted mixture are selected to model the background where B is given by:

$$B = \arg\min_b \left(\sum_{k=1}^{b} w_k > T \right) \tag{2.22}$$

where T is a threshold that specifies the minimum fraction of data to be used for representing the background. T is set in accordance to a trade-off. A small T leads to a small B and to a unimodal background model which requires less computational power. However the model cannot handle repetitive background movements such as the leaves of a tree in wind or similar phenomena. With a large T, the multimodal background scenes can be handled robustly but would require more computational power.

The final decision is to identify the foreground pixels in the scene. A pixel is said to match a distribution if its value X_t does not deviate more than a constant times the variance of that distribution. If the pixel value X_t matches one of the best B distributions which are the background distributions, it is identified in the background. Else it is identified as a foreground pixel. The background image for visualizing the status of the model can be obtained by taking the mean value of the most probable distribution for each pixel.

After the foreground pixels are identified in each frame, they are subjected to some post-processing operations. First, these pixels are morphologically filtered for noise reduction and for filling the gaps and holes in the contours. Then the pixels are segmented to form regions

by using a connected components algorithm. This operation is critical in determining the whole objects which will be necessary later for tracking and behavior identification. Then minimum area filtering is done to discard too small parts. After these standard post-processing operations, an updated foreground image is obtained by union of interiors of the contours of foreground regions.

P. KaewTraKulPong and R. Bowden [8] have improved the original method of Grimson et al. [7]. The presented system in this thesis uses this improved algorithm for background modeling. The original method has the problem of slow learning at the beginning. By modifying the update equations, they obtained a faster and a more accurate adaptation to changing environments. For example, if a foreground object is visible at the instance of initialization, there will be a Gaussian distribution with unity weight to represent the object. After the object moves and the background color is visible in the scene, it takes $\log_{(1-a)}T$ frames for the background color to be identified as a background color for the model and $\log_{(1-a)}0.5$ frames for it to be the dominant color for the model even if the same background color is visible in all period. In addition, ρ is too small and results in too slow adaptations in mean and covariance matrices parameters which cause problems in proper operation in time. Cutting out the likelihood term from ρ is a solution for this problem. P. KaewTraKulPong and R. Bowden [8] start estimation of the Gaussian mixture model by expected sufficient statistics update equations and then use L-recent window version after the first L samples are processed. At the beginning, the expected sufficient statistics update equations estimate accurately till all L samples are obtained. Then L-recent window update equations give priority over recent data so that the changes in the scene can be handled. The update equations for online expectation maximization algorithms by expected sufficient statistics and for the L-recent window are shown respectively as follows:

Sufficient Statistics:

$$\hat{w}_k^{N+1} = \hat{w}_k^N + \frac{1}{N+1}(\hat{p}(w_k \mid x_{N+1}) - \hat{w}_k^N) \tag{2.23}$$

$$\hat{\mu}_k^{N+1} = \hat{\mu}_k^N + \frac{\hat{p}(w_k \mid x_{N+1})}{\sum_{i=1}^{N+1} \hat{p}(w_k \mid x_i)}(x_{N+1} - \hat{\mu}_k^N) \tag{2.24}$$

$$\hat{\Sigma}_k^{N+1} = \hat{\Sigma}_k^N + \frac{\hat{p}(w_k \mid x_{N+1})}{\sum_{i=1}^{N+1} \hat{p}(w_k \mid x_i)}\left((x_{N+1} - \hat{\mu}_k^N)(x_{N+1} - \hat{\mu}_k^N)^T - \hat{\Sigma}_k^N\right) \tag{2.25}$$

L-recent Window:

$$\hat{w}_k^{N+1} = \hat{w}_k^N + \frac{1}{L}(\hat{p}(w_k \mid x_{N+1}) - \hat{w}_k^N) \tag{2.26}$$

$$\hat{\mu}_k^{N+1} = \hat{\mu}_k^N + \frac{1}{L}\left(\frac{\hat{p}(w_k \mid x_{N+1})x_{N+1}}{\hat{w}_k^{N+1}} - \hat{\mu}_k^N\right) \tag{2.27}$$

$$\hat{\Sigma}_k^{N+1} = \hat{\Sigma}_k^N + \frac{1}{L}\left(\frac{\hat{p}(w_k \mid x_{N+1})(x_{N+1} - \hat{\mu}_k^N)(x_{N+1} - \hat{\mu}_k^N)^T}{\hat{w}_k^{N+1}} - \hat{\Sigma}_k^N\right) \tag{2.28}$$

In summary, Mixture of Gaussians background model is superior to other background models for its efficiency and analytical form. This method can handle the scenes with lighting changes by adapting the parameters of the Gaussians and also the multi-modal scenes caused by shadows, moving branches, monitors, and other troublesome features. The method has advantages such as quick recovering when background reappears and an automatic pixel-wise threshold. These factors make this popular and effective method an essential part of our system. A drawback of this method is the assumption of independence of the neighboring pixels and reliance on only the difference between current pixel value and its past values. With the increasing processing

power of systems this method will be run with a better performance using larger images and larger number of Gaussian distributions in the mixture. Using a full covariance matrix will also raise the performance. Adding prediction to each Gaussian in the mixture may also result in better adaptation for this method.

2.1.4 Eigen-Backgrounds Method

As described in the previous sections, most of the background modeling methods identify moving objects by comparing the captured images with an obtained reference frame that represents the static structure of the scene. The reference frame continuously adapts to the various lighting conditions in order to detect the moving objects effectively. The popular method of Mixture of Gaussians builds and updates a multimodal representation of the background for each pixel. However it fails to take into account the substantial degree of correlation between neighboring pixels.

In the method of Oliver and Pentland [9], an eigenspace that models the background scene is built adaptively. The eigenspace model describes the change of pixel values in the scene due to variations in lightning. The eigenspace model is built by collecting a training set of N sample images { I_i for i=1,…,N }, computing the mean background image μ by simply averaging or adaptively and the covariance matrix C_b. The covariance matrix C_b can be diagonalized by eigenvalue decomposition:

$$L_b = \phi_b C_b \phi_b^T \tag{2.29}$$

where Φ_b is the eigenvector matrix of the covariance and L_b is the diagonal eigenvalue matrix. By Principal Component Analysis the dimensionality of the space is reduced by keeping the M eigenvectors

(eigen-backgrounds) with the M largest eigenvalues to obtain Φ_M matrix.

After the eigen-background images are obtained and stored in the matrix Φ_M and the mean background image μ is found, each input image I_i is projected in the subspace spanned by the eigen-background images as:

$$B_i = \phi_M X_i \tag{2.30}$$

where $X_i = I_i - \mu$ is the mean normalized image vector.

Since the subspace only represents the static parts of the scene, by comparing the input image and the projected image and by thresholding the difference between them, the foreground objects that are visible in the scene are identified:

$$D_i = \left| I_i - B_i \right| > Thr \tag{2.31}$$

where D_i denotes the difference and Thr denotes the threshold.

Under the assumption that the moving objects are not visible in the same location of the scene in N observations and they are relatively small, they will not contribute significantly to the model. The static parts of the scene which are generally the background elements are described efficiently as a combination of eigenbasis vectors by the eigenspace model whereas the parts with moving objects cannot be effectively described by the model. Thus, the eigenspace can robustly model the probability distribution function for the background scene, but not for the parts with moving objects.

By adaptively performing eigen-background subtraction, changes in illuminations can be handled effectively. This method has less computational load than Mixture of Gaussians method. However, Mixture of Gaussians method has superior performance with its multimodal behavior.

J. Rymel, J. Renno et al. [10] proposed an adaptive method based on the eigen-Background model by extending the work of Oliver and Pentland [9]. This method continuously learns the covariation within a sequence of captured frames by using Principal Component Analysis to generate the eigen background. The eigen-background model is built and adapted online evolving the parameters and dimensionality rather than acquiring the necessary training set. As new frames are captured, a reference frame is obtained using a subsample of the captured frames. This is an extension to the original method of eigen-backgrounds.

During initialization, with every captured frame the dimension of the subspace is incremented. Incrementing the dimensionality is required in order to have an accurate model to describe the captured frames. This task is performed continuously until the dimension reach to a specified optimal number since there is not enough data for the model to be effective during initialization. Selecting a larger number results in more robustness but requires more computations.

In Rymel et al.'s method [10] the frames are divided into grid of neighborhoods and the statistical variations within each of them are learned continuously in order to obtain a stochastic representation of the background. As new frames are captured, the statistical model for the neighborhoods is updated. Detection is achieved by thresholding gray level differences against estimates of the gray level variance at each pixel.

Since the Eigen-Background method of Rymel et al. benefits from covariability of pixel intensities and estimates the background efficiently with few subsamples, it has the advantages of the requirement for less processing power and less memory during execution. This is accomplished by online incrementation of the subspace dimension and the adaption of subspace during eigen analysis algorithm. However, the previously described state-of-the-art method, Mixture of Gaussians, has the multimodal characteristic and can handle the background scenes with various types of changes and also rapid variations of light accurately.

2.1.5 Comparison of Background Modeling Methods

The performance evaluation of the background modeling methods is accomplished by comparing the background estimate outputs with the manually derived ground truth. A rectangular bounding box covering the true foreground pixels is positioned manually. The percentages of successfully identified foreground and background pixels are represented by the terms detection rate and the specificity respectively. The performance measure is obtained by multiplying these two percentages.

There are many publicly available benchmarking data that are disseminated for evaluation of surveillance systems. Two of them, PETS and DIRC Datasets, which have different scene conditions, are used in Rymel [10] et al.'s work for benchmarking the background models described in the thesis:

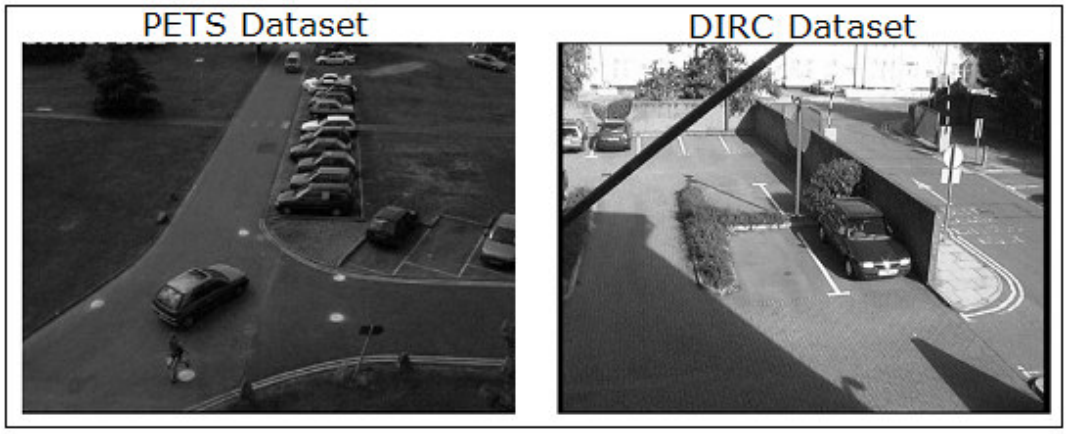


Figure 2.2: PETS and DIRC datasets

The first dataset is PETS 2001. This dataset is created with a stationary camera located at a high altitude point with a steep look-down angle. In the scene there exist a few small and distant objects and a few occlusions occur. The other data used is the in-house DIRC dataset which also has RGB color format. There are constant variations in illumination in time.

In Figure 2.3, which is taken by Rymel et al.'s work [10], the performances of Pentland's Eigen-Background Model, Grimson's GMM method and Rymel's method for a frame set of PETS dataset are illustrated. Ignoring the parts of the plots where metric is zero which occur when there is no object in the scene, it can be analyzed that the Pentland's method has worse performance than the other two methods because it cannot adapt to changes in scene. The method of Rymel and the method of Grimson can handle changes in the scene since they have adaptive behavior.

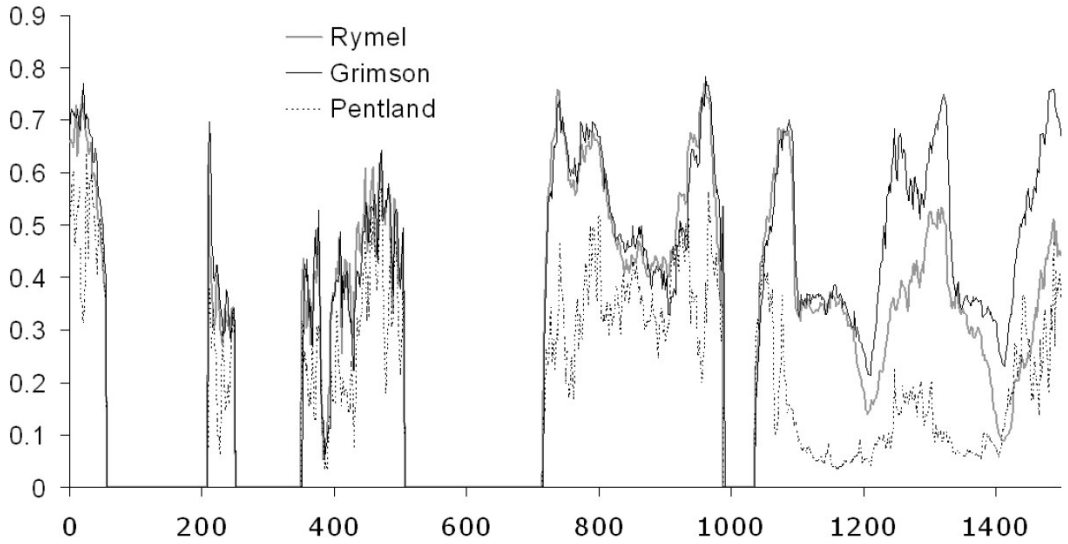


Figure 2.3: Performance results of each method [10]

A better evaluation of the performance is accomplished by using both datasets and averaging the results. In Table 2.1, which is given in Rymel et al.'s work [10], four previously discussed background modeling methods are evaluated. The evaluated methods are Temporal Averaging, Wren's Single Gaussian Distribution method [6], Grimson's Mixture of Gaussians method [7] and Rymel's Eigen-Background method [10].

Table 2.1: Performance of discussed background modeling methods [10]

	Background Modeling Methods			
Dataset	Temporal Smoothing	Wren's SGM	Grimson's MGM	Rymel's Eigen-BG
PETS Dataset	0.11	0.13	0.28	0.22
DIRC Dataset	0.15	0.18	0.33	0.31

Analyzing Figure 2.3 and Table 2.1, it is observed that the method of Rymel and the method of Stauffer and Grimson have far more performance than the other two methods. Rymel's method is more efficient in memory usage; however Stauffer and Grimson's state-of-the-art method, Mixture of Gaussians, has a better performance. This method is the only one with the multi-modal ability within the analyzed methods which allows it to handle many variations in the scene.

There are also some other methods of background modeling where a statistical representation of the background is built. These models are called non-parametric methods. These methods estimate the probability distribution function directly from the samples without any assumptions to any distribution such as a Gaussian distribution since For instance, a Gaussian assumption for pixel intensity distribution may not always hold. The non-parametric methods avoid having to use a limited model and estimating its parameters. A popular method is non-parametric method of background modeling by Parzen Density Estimation [11]. In this method, a kernel estimator is used for each pixel. The disadvantage of non-parametric methods is the high computational load; therefore it is not practical to use them in a real-time embedded system.

2.2 Detection Algorithm

The mid-step between Background Modeling and Tracking tasks is the Detection task. The presented system assumes a single object in the scene; therefore the object is identified by foreground image directly. Multiple object detection is a more challenging problem and it is out-of-scope of this thesis. So we do not give a detailed survey on object detection in this thesis.

In order to detect the moving object in the scene, the foreground scene which is obtained by MGM method is analyzed. The proposed system

can detect only a single moving object. For automatically detection of the object, the region of interest on the scene is obtained and re-initialized when necessary during operation.

2.2.1 Finding Region of Interest

The system simply detects the foreground blob in the captured frames and applies size filtering for the detected object. The object size should not be very large or too small. The system continues to perform background modeling and checks the location and size of the blob until the object is totally visible in the field of view of camera (See Figure 2.4). Then the location of the object is selected as the region of interest for finding the features to be tracked.

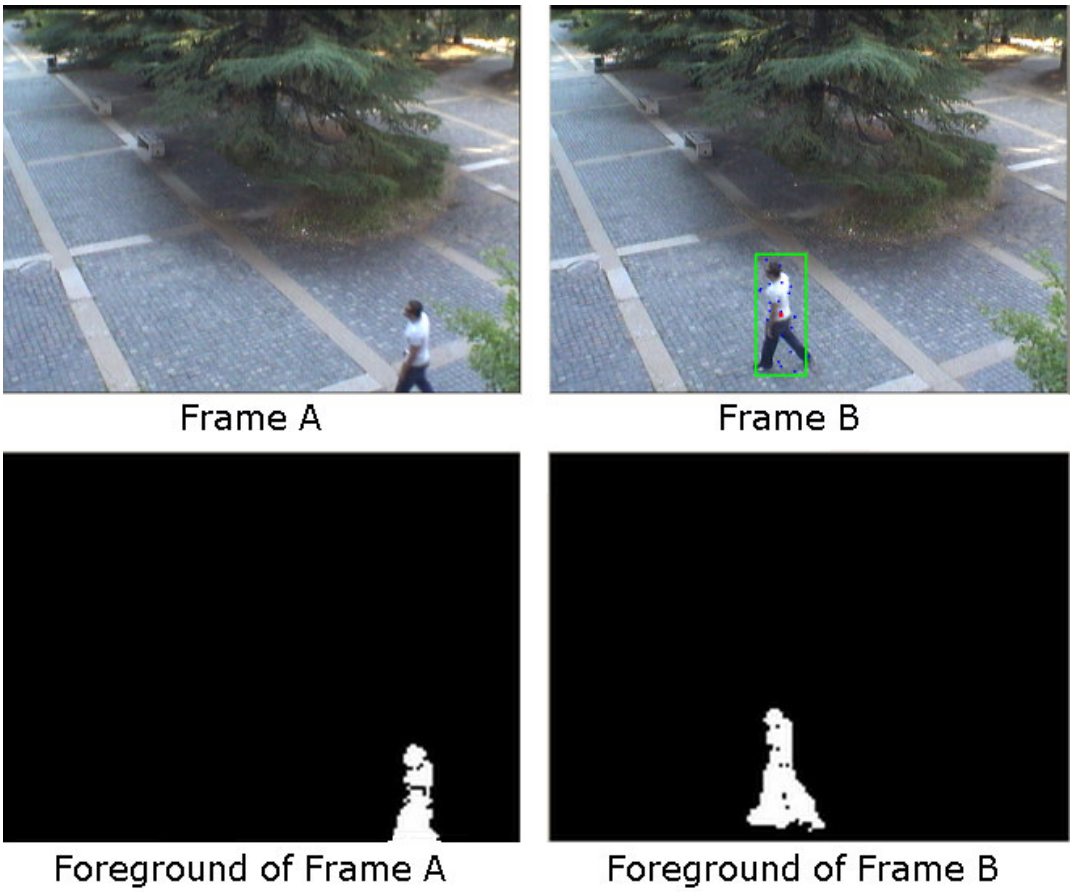


Figure 2.4: Features are not extracted until the whole object is visible

2.3 Object Tracking

Object tracking, which is performed after background modeling and object detection, is one of the most critical tasks of the proposed system in this thesis. Tracking is the process of finding the location of an object or several objects in an image sequence over time.

A successful tracking method needs to be robust, fast and accurate enough to handle cases in which an object is moving rapidly or even when the capturing rate is low. The tracking algorithms generally track a moving object in the scene by identifying the tracked object in consecutive frames using its characteristics. There are four main approaches for object tracking that use different characteristics of the objects: region-based, active-contour-based, model-based and feature-based. There can be also hybrid approaches that integrate some of these main approaches. All of these approaches have advantages and disadvantages when compared to each other. In this thesis, a single object, which is identified using background and foreground segmentation, is tracked using the feature-based approach which will be described in detail in Section 2.3.4.

2.3.1 Region-Based Tracking

In Region-Based approach, tracking is performed according to the variation of the image regions corresponding to moving objects in the scene. The regions or blobs (binary large object) are formed by connecting group of pixels as the background is modeled and the foreground pixels are identified. Then the regions are detected in following captured frames.

Region-Based approach has problems with handling scenes with occlusions, the intersecting objects in the scene are grouped as a large

blob in foreground scene. Therefore, this approach cannot handle complex and crowded scenes.

In order to perform tracking of people accurately in case of occlusions, McKenna [12] used an adaptive background subtraction method and three levels of abstraction to identify regions, people and groups during tracking. They identified a person with multiple blobs taking into account the geometrical structure of the human body. A group of people is identified by observing many individual persons together. In this method the blobs can merge and split.

Sindhu and T. Morris [13] used a region based method that tracks objects well in case of occlusions and low resolution, noisy scenes.

Zhao et al. [14], taking into account the similarity of the appearance of people in successive frames, stated that appearance information is more robust than model information. Therefore, by grouping body parts and building an appearance model for the body parts, they analyzed the similarity of these parts in captured frames. In addition, they performed head detection for all captured frames and torso detection when head detection failed to accurately analyze the scene. By using many distinctive characteristics and adaptively weighting them they obtained a reliable system. As they use multiple characteristics of the person, model and track them separately, their method can handle scenes with occlusions by tracking more significant characteristics. Furthermore, the motion of a certain body part is more stable when compared with whole body of a person.

In the literature there are also many methods of Region-Based tracking that use Kalman filtering [15], [16], [17], [18].

Region Based Tracking has the disadvantage of computational complexity as in these methods a window is matched with all candidate windows in captured frames. The other problem is that the intersecting objects in the scene are grouped as a large blob in the foreground in scenes with occlusions. It cannot reliably handle occlusion between objects [19]. Therefore this approach cannot handle complex and crowded scenes [20]. Furthermore, since these algorithms generally deal with identifying and tracking people, this approach is not suitable for the system proposed in this thesis. It needs to track both people and objects. In addition, these algorithms run on scenes with stationary camera which will require camera compensation, an extra task for the system.

2.3.2 Active-Contour-Based Tracking

In Active-Contour-Based approach, the dual of Region-Based approach, the outlines of objects, the bounding contours, are used for identifying and tracking the objects. The bounding contours are updated dynamically as new frames are captured [21], [22], [23], [24], [25]. In these methods the aim is to extract the shapes of objects and to describe the objects more simply and more efficiently than the region-based methods.

Sun and Haynor et al. [26] proposed an algorithm to automatically detect the deformations on an object and to adjust the object contour to match the real object boundary without breaking the object into several pieces while tracking.

The objects are identified and tracked using VSnakes algorithm in [27]. Rather than using the energy of the contour, a differential active contour energy that reflects the succeeding contour configurations is used. In [28] motion information is used in order to remove background clutter by the proposed tracking system which is based on

active contours. After finding an approximation for the position of the tracked object, the active contour position is found by minimizing the energy function according to the predicted position of the object and the edge map near the predicted object contour. In [29] a deformable model that combines point tracking and edge tracking information is described.

Active-contour based tracking describe the objects more simply, more efficiently and with lower computational complexity than the region-based tracking [25]. On the other hand, this approach is very sensitive to initialization of tracking which makes it difficult to automatically start tracking. Furthermore, the inability to identify and track partially occluded objects remains also in this approach as in the Region-based approach. The tracking could be done even in case of partial occlusion if a separate contour could be initialized for each object. Another disadvantage of this approach is that the tracking precision is limited at the contour level [20].

2.3.3 Model-Based Tracking

In Model-based approach, 2D or 3D projected object models are developed offline using prior knowledge of the object, usually with computer vision techniques or manual measurement. Afterwards the object is tracked by matching these projected object models to the image data [25]. This property allows the recovering of trajectories and models with high accuracy for a small number of objects, and even to address the problem of partial occlusion. Model-based algorithm performs well under occlusion but has a drawback of high computational cost.

Model-based approach is also used for human body tracking which is known as analysis by synthesis. Algorithm is used in a predict-match-update style. As in the object tracking, pose of the model for the next

frame is predicted according to prior knowledge and tracking history. Then the human body is tracked by matching the projected object to the image data. The difference for the human-body tracking is the knowledge of the general body model and the motion constraints of the human.

A sub approach for this algorithm is looking for an object in a frame which conforms to some sort a model of what is to be tracked. This method can be efficient depending on the complexity of the model to be tracked and how easily it is to search for this model in the frame. This approach is also efficient in cluttered scenes where isolating individual objects can be difficult. An example of this is in the system developed by Tweed and Calway [30] for tracking birds flying in large flocks. Another disadvantage of this approach is that it can track only a predetermined class of object.

2.3.4 Feature-Based Tracking

Among the many discussed approaches for tracking in video, the Feature-Based approach is the main one to be robust to partial occlusions. In this approach, the objects are tracked by extracting their features such as intensity, color, edges or corners and by finding the correspondence between these features in successive frames rather than tracking the objects as whole.

In the literature, there are various feature-based algorithms proposed which can be classified into three groups depending on the characteristics of the used features as: global feature-based algorithms, local feature-based algorithms and dependence-graph-based algorithms.

Global feature-based algorithms use features such as areas, centroids, perimeters or colors. Polana and Nelson [31] used centroids as the

features. The system they proposed tracks a person by following the centroid of the moving pixels. The tracking algorithm smoothly tracks the person even in case of occlusions.

In local feature-based algorithms, features such as segments, curve segments, and corners are used. In Beymer et al.'s work [32]; the most salient features are selected depending on the nature of the scene in order to have good performance in many conditions.

Dependence-graph-based algorithms use variety of distances and geometric relations between features. In Fan and Medioni et al.'s work [33], objects are described in terms of their surfaces. The surface of an object is described by segmenting it into surface patches and the complete description consists of the description of each patch separately, and their interrelationships. Dependence graph-based algorithms are not suitable for use in real-time tracking because searching and matching of graphs involves too many computations and consumes too much time.

Algorithms that belong to these three groups can also be combined. Jang et al. [34] dynamically built an active template that characterizes regional and structural features of an object depending on color, edge, shape, and texture information. The tracking of the moving object is accomplished by using motion estimation or Kalman filtering in order to minimize an energy function during feature matching.

Feature-based algorithms are generally fast enough which makes them suitable for real-time tracking and tracking of multiple objects. A distinct advantage of feature-based tracking algorithms is the robustness to partial occlusions. The partial occlusion can be handled by some methods such as using local features or by analyzing motion of object, etc. In case of partial occlusion, some of the features of the

moving object remain visible, so it may be tracked by selecting the most salient features according to the scene parameters. This approach can be used in many illumination and scene conditions. However, these algorithms are sensitive to image variations and they cannot recover 3D pose of objects.

One of the most developed algorithms of Feature-based tracking approach is the Kanade-Lucas-Tomasi (KLT) Feature Tracker. Lucas and Kanade [35] proposed a method for image registration which uses special intensity gradients of images to perform matching for use in stereo vision systems. Then, based on [35], Tomasi and Kanade [36] presented a feature tracker using sum of squared intensity differences over the windows.

The proposed system in this thesis uses Pyramidal Implementation of the Lucas Kanade Feature Tracker [37] as the tracking algorithm for features which are actually the corners (Feature selection is described in detail in Section 2.3.6). It is a very popular, robust, and fast algorithm making it a proper choice for use in real-time applications or on an embedded system. This algorithm can find the correspondences of features in sub-pixel accuracy. The pyramidal structure, making calculations in different resolutions, enables handling of large pixel motion. This algorithm has adequate performance of tracking objects in a scene with moving camera. The details of the algorithm are described in Section 2.3.5.

2.3.5 Pyramidal Lucas Kanade Feature Tracker

Lucas Kanade Feature Tracker is used in feature tracking and is based on optical flow algorithm. The goal of this tracking algorithm is to find the location of a pixel on the latter frame, while knowing the coordinates of the same pixel on the initial frame.

In this approach, the object is assumed to move within a limited range between the frames, so rapid motions are hard to track. Let I and J be successive frames, u=(u_x, u_y) be the image point on the first frame, *d*=(d_x, d_y) be the displacement vector of the point to be tracked, and image neighborhood size be $(2w_x+1)$ pixels horizontally and $(2w_y+1)$ pixels vertically. Then the image velocity *d* as being the vector that minimizes the residual function epsilon is defined as follows:

$$\varepsilon(d)=\varepsilon(d_x,d_y)=\sum_{x=u_x-w_x}^{u_x+w_x}\sum_{y=u_y-w_y}^{u_y+w_y}(I(x,y)-J(x+d_x,y+d_y))^2 \qquad (2.32)$$

w_x and w_y numbers are assumed to be integers up to 7, so the maximum displacement is assumed to be 15 pixels. However, the pyramidal approach handles more displacements that can be caused by rapid movements. The pyramids are formed by filtering and sub-sampling the images in a recursive manner starting from the 0^{th} level, which is the original level, to the coarsest level. Then, the displacement vectors are found iteratively in upward from the coarsest level to the original level. At each level, displacement vector is calculated by maximizing a correlation measure over a small window. The pixels at the image borders are handled by mirroring and using replicas of the rows and columns of the images instead of the out-of-border parts which are not defined.

2.3.5.1 Pyramidal Feature Tracking

For L = 0, …., L_m define $u^L=\begin{bmatrix}u_x^L & u_y^L\end{bmatrix}$, the corresponding coordinates of the point u on the pyramidal images I^L. Then u^L is computed from as:

$$u^L=\frac{u}{2^L} \qquad (2.33)$$

At each resolution level, past results are used as initial guesses. Let the initial guess be $g^L = \begin{bmatrix} g_x^L & g_y^L \end{bmatrix}^T$. Then, in order to compute the optical flow at level L, it is necessary to find the residual pixel displacement vector d^L that minimizes the new image matching error function ε^L as:

$$\varepsilon^L(d^L) = \varepsilon^L(d_x^L, d_y^L) = \sum_{x=u_x^L-w_x}^{u_x^L+w_x} \sum_{y=u_y^L-w_y}^{u_y^L+w_y} (I^L(x,y) - J^L(x+g_x^L+d_x^L, y+g_y^L+d_y^L))^2 \tag{2.34}$$

Observe that the search window has a constant size at each level. Since the initial guess vector g^L is used to pre-translate the image patch, the residual flow vector d^L comes out to be small and easy to be computed through a standard Lucas Kanade step. After d^L is calculated, the result is propagated to the next level L-1 by passing the new initial guess g^{L-1}:

$$g^{L-1} = 2(g^L + d^L) \tag{2.35}$$

The next level optical flow residual vector d^{L-1} is computed in the same way. This procedure goes on until the highest resolution is reached. At first the initial guess for level L_m is zero since no initial guess is available at the deepest level of the pyramid:

$$g^{L_m} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T \tag{2.36}$$

Solution vector d is found after the final optical flow calculation:

$$d = g^0 + d^0 \tag{2.37}$$

or *d* can be expressed as the sum of the residual optical vectors d^L:

$$d = \sum_{L=0}^{L_m} 2^L d^L \qquad (2.38)$$

2.3.5.2 Tracking features close to the boundary of the images

Points that are close to the boundaries of the frame need to be processed since some part of their integration window lies outside of the frame. For an integration window having size $(2w_x+1)\,\mathrm{x}\,(2w_y+1)$, forbidden band of width w_x and w_y occurs around the image. In the pyramidal implementation, this corresponds to an effective forbidden band having width $2Lmw_x$ and height $2Lmw_y$. The size of this forbidden band has significant for large values of Lm. For example, for Lm=3, width value becomes $8w_x$. Notice that w_x takes integer values up to 7, which makes the size of the forbidden band 56 pixels around the image. This problem can be solved by calculating the summations in the equations only for the valid portion of the image neighborhood which are within the frame, i.e. for valid entries of $I_x(x,y), I_y(x,y)$ and $\delta I_k(x,y)$.

2.3.5.3 Declaring a Feature as Lost

There are two cases in the algorithm that declare a feature as lost:

1. A feature is declared as lost when the feature point falls out of image boundaries. If the point $p = \begin{bmatrix} p_x & p_y \end{bmatrix}^T$ (the center of the neighborhood) falls outside of the image I^L, or if its corresponding tracked point $\begin{bmatrix} p_x + g_x^L + v_x^{k-1} & p_y + g_y^L + v_y^{k-1} \end{bmatrix}$ falls outside of the image J^L, it is reasonable to declare the point as lost, and not to continue to track it.
2. A feature is declared as lost when the image patch around the feature point varies too much between the successive frames *I* and *J*. This condition usually occurs due to occlusion.

2.3.6 Extracting and Handling Features

After the region of interest is defined, the features on this region are extracted for use in tracking. Harris corner detector is used in the proposed system. Using Harris detector ensures that the spatial gradient matrix, which is used in Lucas Kanade Tracker Implementation, has two eigenvalues and therefore it is invertible. This is one of the reasons for using Harris corner detector. Minimum eigenvalue method can also be used. In this method, a pixel is selected if eigenvalues of spatial gradient matrix are larger than a threshold. As even the smaller eigenvalue is larger than a threshold, the spatial gradient matrix is also well conditioned.

Harris corner detector is a popular detector which is resistant to rotation, illumination changes and noise [38]. However it is not resistant to scaling, since a corner may be observed as an edge when magnified or a smooth edge can be identified as a corner when it is seen in a smaller scale. When zooming is performed in order to keep the object size same on the video frames as it moves, the scaling effect will be less significant.

Around a corner, there is a large intensity change in all directions. The local autocorrelation function of the signal, which is given by Equation 2.39, is used for measuring the local intensity changes in different directions.

$$C(x,y)=\begin{bmatrix} \sum_R w_R I_x^2 & \sum_R w_R I_x I_y \\ \sum_R w_R I_x I_y & \sum_R w_R I_y^2 \end{bmatrix} \tag{2.39}$$

I_x and I_y are the partial derivatives of the image and (x, y) are the points in a Gaussian window w_R centered on (x, y) in the region R. The eigenvalue of the autocorrelation function on the direction of intensity

change is large. Therefore, for a corner both eigenvalues are large. A corner response function is defined by:

$$M = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \qquad (2.40)$$

where λ_1 and λ_2 are eigenvalues of autocorrelation function and k is a constant that can be used to vary the number of detected corners. The result of this function is large for a corner, negative for an edge and small for a flat region. Using a threshold the local maximas of this function is used to identify the corners. Finally, the corners which are located very close to some other stronger corners are discarded.

Figure 2.5: Harris Corner Detector

After the strong features are identified on the region of interest, they are tested for consistency by the system. In addition to the analysis of features by Lucas Kanade algorithm, with every captured frame, the motions of features are analyzed for consistency. The motion of each feature is compared with the motion of mean location of the features. If the displacements of a feature and the mean of features are not in the same direction and if there is a large difference between them, the feature is eliminated. During tracking as the number of valid features becomes less than a specified number, the system stops the movement

of the camera, re-initializes the background model, finds the foreground object, finds the new features and continues tracking. Frame difference is used for the re-initialization purpose during tracking since it is fast and simple whereas updating the Gaussian model is not practical while the object is moving in the scene and the system can miss the moving object while waiting for the background model to be re-initialized. By using re-initialization, the tracking operation can continue for larger time periods. Another advantage of handling features is that at the first initialization of features, some of them may be located in the background. With this operation, those features that are on the background are also discarded. Another modification on features is that the features which are located far away from the mean location of the pixels are discarded and not used in tracking. The last modification is that if the number of features reduces so rapidly to zero which can occur when the object is totally occluded or when the object leaves the field of view, the tracking operation is stopped. These modifications lead to more stable tracking.

CHAPTER 3

HARDWARE OF THE SYSTEM

The hardware of the system proposed in this thesis consists of a digital image/video processing board, Texas Instruments DM642 Evaluation Module (EVM) and an analog pan-tilt-zoom camera. In this application, the digital video processing board accomplishes a video loopback and analysis task. It acquires the analog video which is captured by the analog camera, processes it, and outputs the modified video to the monitor. In order to track the moving object in a wide operation range, the board controls the pan, tilt and zoom properties of the camera. The control commands are sent by the serial port. There are many camera control protocols which are dependent on the model of the camera being used. These protocols provide a two-way communication between the processing unit and the connected cameras.

Figure 3.1: Texas Instruments DM642 Evaluation Module

The main processing unit of the system is Texas Instruments DM642 EVM which is a low-cost, high performance video imaging development platform designed to jump-start application development and evaluation of multi-format digital applications.

The other important unit of the system is the pan-tilt-zoom camera which is used to capture composite video and track the detected objects as well. The main features and functions of the video processing board and the camera are explained in detail in Sections 3.1 and 3.2 respectively.

3.1 Overview of TI DM642 Evaluation Module

The most important part of DM642 EVM is the TMS320DM642 digital signal processor (DSP). Digital signal processors have an important role in a wide range of video and imaging applications such as computer vision, medical imaging, security, digital cameras, and a large number of consumer applications driven by digital video processing. The importance of multimedia technology applications is recognized by microprocessor designers. As a result of this fact, the number of special-purpose multimedia processors is becoming more popular.

Multimedia applications can be generally characterized by requirements for processing flexibility, complicated algorithms and high data rates for both input and output. Very Long Instruction Word (VLIW) architecture is one of the processor architectures to overcome these difficult requirements of multimedia applications. VLIW processors can exploit instruction level parallelism (ILP) in programs [39]. TMS320DM642 is one of the devices that is based on VLIW architecture and seems to be a good choice for real-time applications.

DM642 DSP is based on the C64x CPU, which is part of the C6000 DSP family of TI. The DM642 integrates a number of peripherals to address

the development of video and imaging applications, including three configurable video ports capable of glueless video input, video output, or transport stream input. These video ports can support BT.656 video I/O, HDTV Y/C I/O at up to 10-bits per component [40]. The details of the architecture of DM642 are given in the following sections [41].

DM642 core is based on the key features such as VLIW architecture, 2-level memory/cache hierarchy, and EDMA engine that are very important for computationally intensive video and image applications. Figure 3.2 shows the block diagram of the processor.

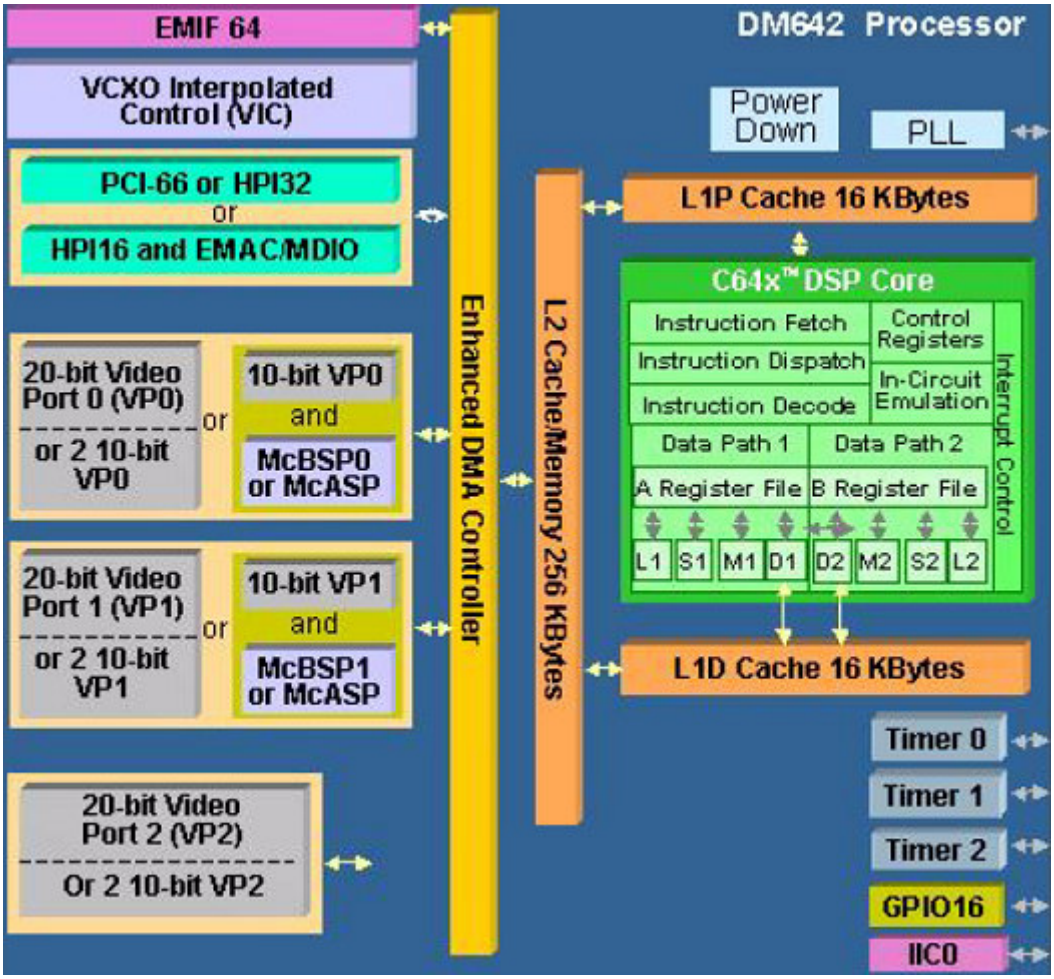


Figure 3.2: DM642 DSP Block Diagram [41]

On DM642 devices, the CPU interfaces directly to dedicated level-one program (L1P) and data (L1D) caches of 16Kbytes each. These caches operate at the full speed of CPU access. A second level unified L2 program/data memory provides flexible storage. One configuration for L2 is entirely mapped SRAM. The other configurations have both SRAM and a 4-way set associative cache of various sizes. Mapped SRAM can be used for streaming video data and critical sections of code such as interrupt service routines. Cache is useful for most of the program and data structures. Figure 3.3 shows L1 cache for DM642 and Figure 3.4 shows the partitioning of 256Kbytes of internal memory into L2 cache/ram in different configurations.

Figure 3.3: DM642 L1 Cache [41].

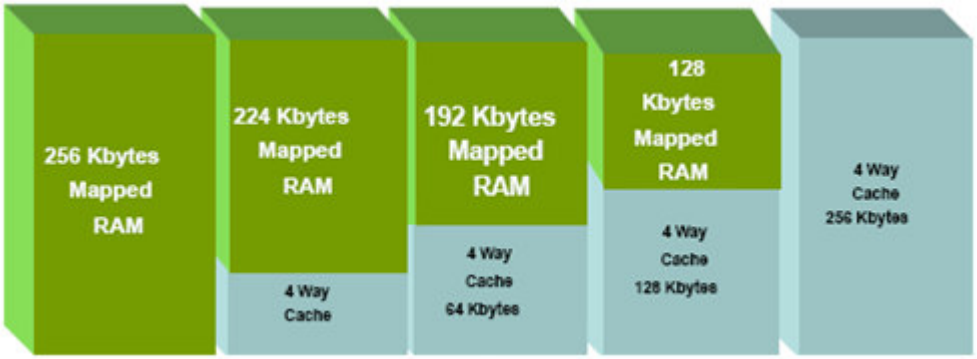


Figure 3.4: Partitioning internal memory into L2 cache/ram

The C64xx family of DSPs has a large byte addressable address space. Program code and data can be placed anywhere in the unified address

space. Addresses are always 32-bits wide. Portions of internal memory can be remapped in software as L2 cache rather than fixed RAM.

There are two methods for transferring data from one part of the memory to another: using CPU and using DMA.

If a DMA is used then the CPU only needs to configure the DMA. While the transfer is taking place, the CPU is free to perform other operations. The EDMA controller handles all data transfers between the level-two (L2) cache/memory controller and the device peripherals on the DM642 DSP. These data transfers include cache servicing, user-programmed data transfers, and host accesses. The EDMA provides the ability to transfer data with zero overhead. It is clear that the EDMA and CPU operations can be independent. However, if the CPU and EDMA both try to access the same memory location, arbitration will be performed by the program memory controller. This should also be taken into account that the following conditions may limit the performance:

1. EDMA stalls when there are multiple transfer requests on the same priority level.
2. EDMA accesses to L2 SRAM with lower priority than the CPU.

DM642 DSP is based on the C64x CPU which includes special instructions to increase the performance of video and imaging processing. With these properties DM642 meets the needs of video and imaging application development. Also, the RISC-like instruction set and extensive use of pipelining in C64x allow many instructions to be scheduled and executed in parallel. A high performance two-level cache design allows the CPU to operate at the maximum rate and the existence of EDMA provides us to transfer data with zero overhead.

The video port peripheral of DM642 can operate as a video capture port, video display port, or transport stream interface (TSI) capture port. The EVM captures both composite and S-video signals and can display composite, S-video, and RGB signals. In this thesis work, the captured and displayed signals are both composite.

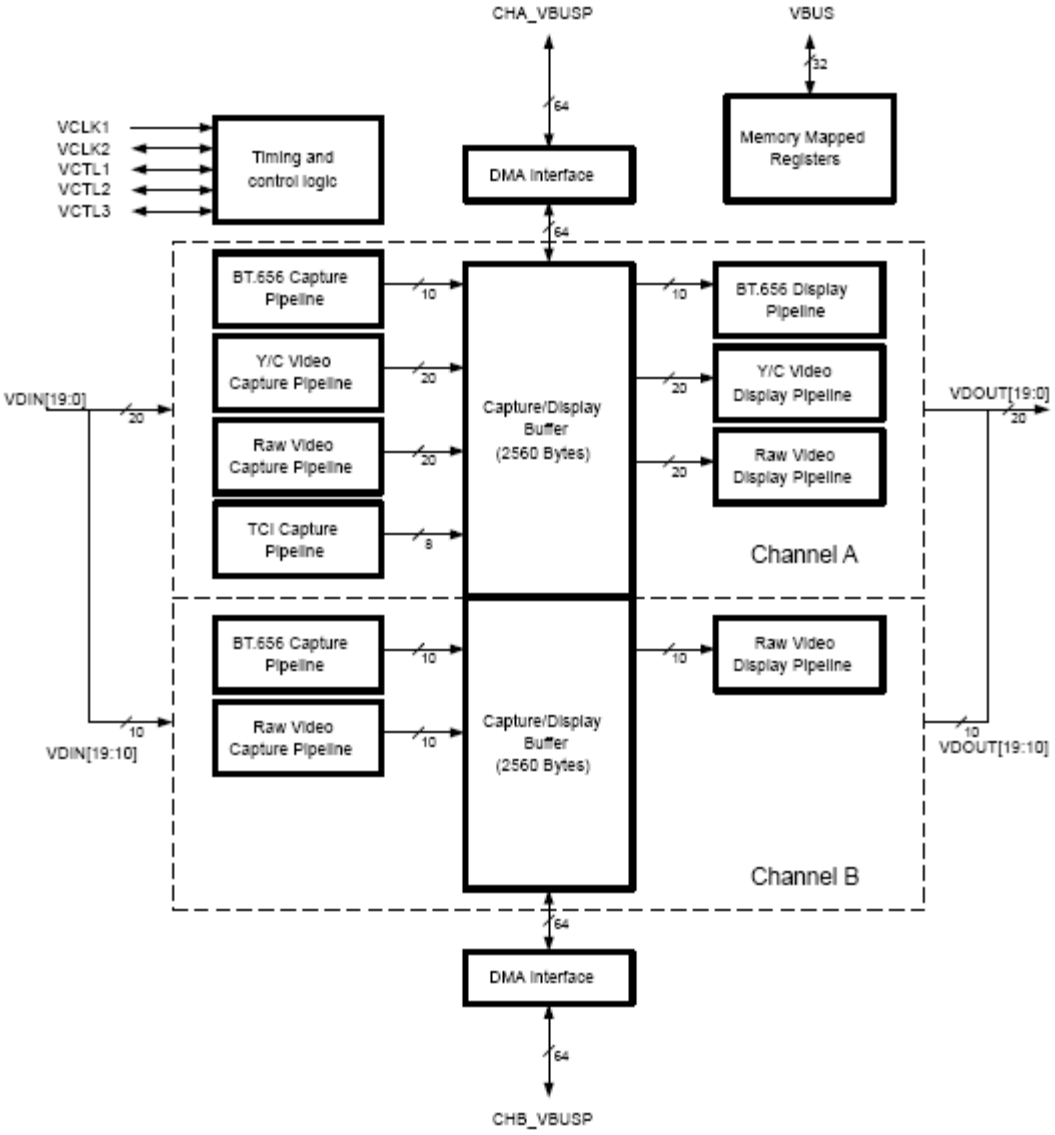


Figure 3.5: Video Port Block Diagram

DM642 EVM has two multichannel buffered serial ports (McBSPs) interface to a variety of standards. McBSPs are synchronous serial ports. These multi-channel ports in combination with the flexible asynchronous interface provide a glueless connection to a variety of products such as the PTZ camera used in this thesis.

Furthermore, DM642 EVM has 32 MB SDRAM external memory, LEDs, GPIOs, PCI, JTAG, audio, ethernet and daughter card interfaces, which can be used in a variety of applications.

3.2 Pan-Tilt-Zoom Camera

In the proposed system, the composite video capturing is performed by using an analog camera. The camera, with integrated pan-tilt-zoom capabilities, can track an object in a wider range which is desired for our purpose. The model used in this work is Sony's EVI-D100/P which is a CCD camera combined with a high-speed and quiet pan-tilt. The range of pan and tilt functions are ±100 degrees and ±25 degrees respectively.

Figure 3.6: Sony EVI-D100P Video Camera

The camera has also features such as auto focus, auto white balance, and automatic exposure to provide fast and stable operation and serial control interface and remote commander to enable camera control.

VISCA is the protocol that is supported by the camera. VISCA provides a two-way communication which enables the EVM to send commands, receive inquiries and get responses about the status of the actions of camera. The interface to the protocol is RS232C with 9600 bps, 8 bits data, 1 start bit, 1 stop bit and non parity. A VISCA command data packet consists of a 1-byte header, maximum 14 bytes of data and a terminator byte of 0xFF. Each command data packet has a corresponding response data packet.

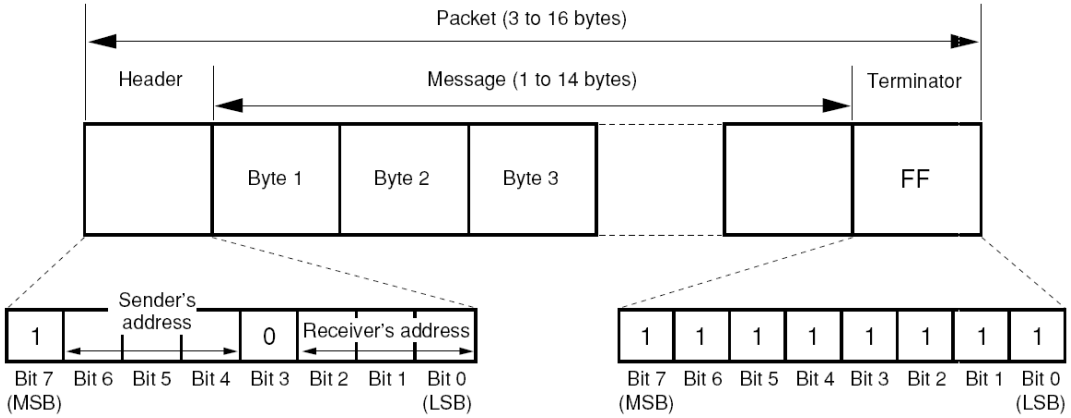


Figure 3.7: VISCA Packet Structure

Implementing this communication protocol allows the control of many features of the camera including pan-tilt-zoom functions. There are many camera control protocols that are dependent on the model used. PELCO-D is the most popular camera protocol which is used by many manufacturers. A packet of this protocol contains 7 bytes with the given format [44]:

Table 3.1: PELCO Packet Structure

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sync Byte	Camera Address	Command 1	Command 2	Data 1	Data 2	Checksum Byte

There are also many other protocols which are not as popular as PELCO-D. They are mostly developed by the manufacturers (only such as Canon Samsung, LG, Bosch, Honeywell, etc) for use in their products and some of these protocols are not publicly available.

CHAPTER 4

SYSTEM IMPLEMENTATION

The hardware used for the implementation of the proposed system and the details of the generated software is presented in this chapter.

4.1 Hardware Setup

The final presented system mainly consists of DM642 EVM and the analog pan-tilt-zoom camera. However, there are other devices that are also used for debugging and development tasks. All devices that are used together and their connections are described in this section. The whole system for the development includes: DM642 EVM, the PTZ camera, XDS560 emulator of Texas Instruments, PC with TI Code Composer Studio (CCS) and a PAL monitor or TV.

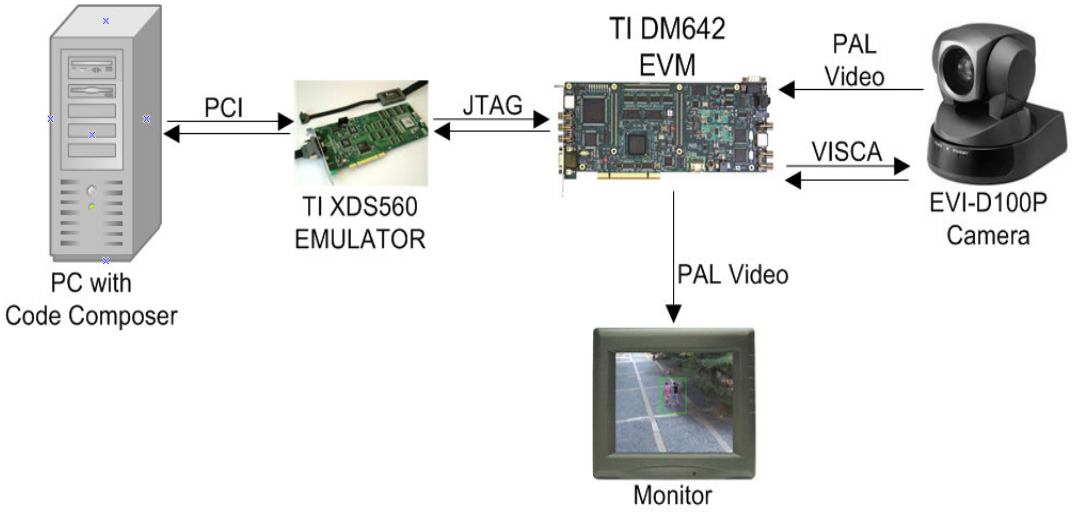


Figure 4.1: Used Hardware for Debugging and Development

Texas Instruments XDS560 is the emulator that is used for debugging tasks while developing the presented system. XDS560 is a PCI-based emulator which is designed to operate as a universal card in a PCI expansion slot of the host PC. The XDS560 supports the full range of standard emulation and debugging capabilities, such as software and real-time hardware breakpoint, single-step execution, loading, inspecting and modification of all registers and memory and benchmarking of execution time of clock cycles [42]. These features are provided with Texas Instruments' Code Composer Studio (CCS) software which is installed in the PC. This software allows compiling, loading, debugging and executing the written codes on the EVM. It also offers the ability to have plug-in applications that both control and visualize emulation data coming from the target device via the XDS560 emulator [43]. XDS560 on the host PC communicates through the JTAG interface to the target DSP, DM642, with its embedded emulation components.

For observing the processed video signal, a PAL monitor is used in the setup. The composite video output of DM642 EVM is used for transmitting the video signal to the monitor or TV.

DM642 EVM and the analog pan-tilt-zoom camera have two connections. The first one is the transmission of PAL video signal to composite video input of EVM642. The second connection is the serial communication of the camera and the McBSP of EVM for camera control by using VISCA protocol (Section 3.2).

The hardware used in the thesis was tested with many tools. The VISCA-RS232C connection was tested and verified with a serial port monitoring software. The JTAG and PCI connections were tested using diagnostic tools of Texas Instruments. Finally, after all devices are

connected, all hardware peripherals were tested and verified by preparing test codes.

4.2 Software Development

Up to this section, the methods of background modeling, feature extraction and tracking and hardware components of the system are introduced. In this section, the architecture of software of the presented system and the implementation details are described.

The software development part consists of two phases. The first phase is to write the code in C++ programming language on PC in Microsoft Visual Studio Environment and test the algorithms and whole code. In the second phase, the written code is ported to embedded platform of DM642, which is the DSP of the system, using Code Composer Studio (CCS) Integrated Development Environment (IDE). The second phase requires some extra coding and work for initializing the DM642 board, creating tasks and communication between tasks, setting up video encoder and video decoder for video capture and display, and serial port for camera control and finally setting up user interface. There are also difficulties such as: many libraries or functions are not available in embedded platform, the memory sources are limited and debugging is more time consuming.

As previously described in Section 4.1, at the hardware level the emulator XDS560 located in host PC communicates with the DM642 EVM through JTAG port. Similarly, at the software level the software CCS which is installed in host PC communicates with DSP/BIOS operating system on the target board. DSP/BIOS system handles the algorithms, tasks and drivers which are executed on the board. In the presented system, the code developed in Windows environment is built on DSP/BIOS by generating threads and ordering them properly. There are five tasks on DSP/BIOS for this system: tsk_idle, tskCapture,

tskControl, tskProcess and tskDisplay. These tasks can communicate with each other using SCOM module of CCS. This module organizes the execution of tasks in a proper order.

tsk_idle is the default task of the processor. Therefore, it is not an executed task.

tskCapture is the task for capturing video signal from the video decoder. In this task, the decoder is configured for capturing composite (PAL) video signal, a capture channel and the SCOM links for receiving and sending are created and allocations are done during initialization. After initialization, the task continuously checks the message from other tasks and when it receives the message, it sends the captured frame to the SCOM queue for processing by EDMA and receives an empty buffer for use in the next loop.

tskControl is the task that is used for user control. The task continuously checks for user commands. Code Composer Studio allows adding menu items and creating user interface to set some parameters of the DSP and to control the code. This can be done by creating General Extension Language (GEL) files. The operation mode of the system is selected by the prepared GEL file.

tskProcess is the most important task of the code since the algorithms are executed within this task. In the initialization part of this task, all variables and buffers that are used in the algorithms are allocated. The peripherals and the McBSP are initialized by Chip Support Library (CSL) of CCS for DM642. CSL provides a C-language interface for configuring and controlling on-chip peripherals. The camera is initialized and connected by commands of VISCA protocol.

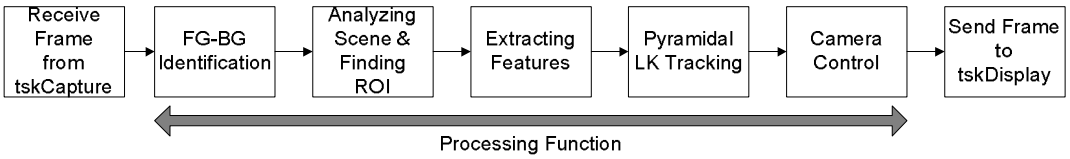


Figure 4.2: Block Diagram of tskProcess

This task at each loop, gets an input buffer from the capture task, processes it and sends back to the display task after processing. The processing function uses down-sampled images since the memory for the embedded system is limited. The EVM has only 32 MB of RAM. In addition, for faster operation YUV color format, which is the default color format of video peripherals, is used. These color components are less correlated than RGB and thus the assumptions of independence in MGM are also valid in the model.

The processing function initiates the MGM method which is explained in Section 2.1.3 when the camera is stationary. The model is updated with each received frame. At a glance, the algorithm is mainly in the form:

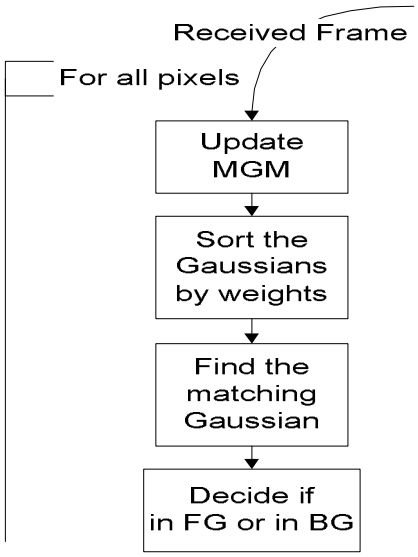


Figure 4.3: Main Parts of MGM algorithm

The parameter K, which is the number of Gaussians, is chosen as 2 depending on the available memory of the EVM. Again, as new frames are received, the foreground scene is identified as described in Section 2.2.1. When a desired foreground object is detected on the scene, the features on the object are extracted as told in Section 2.3.6. Then, the tracker which is given in 2.3.5 starts to operate. For any feature of the object on the first frame, which is denoted by *u*, the Pyramidal LK tracking algorithm estimates the corresponding location *v* on the next frame *J*. The implementation is as follows:

Build pyramid representations of I and J:

$$\{I^L\}_{L=0,\ldots,L_m} \text{ and } \{J^L\}_{L=0,\ldots,L_m} \tag{4.1}$$

Initialization of pyramidal guess:

$$g^{L_m} = \begin{bmatrix} g_x^{L_m} & g_x^{L_m} \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T \tag{4.2}$$

for L = Lm down to 0 with step of -1

Location of point u on image I^L:

$$u^L = \begin{bmatrix} p_x & p_y \end{bmatrix} = \frac{u}{2^L} \tag{4.3}$$

Derivative of I^L with respect to x:

$$I_x(x,y) = \frac{I^L(x+1,y) - I^L(x-1,y)}{2} \tag{4.4}$$

Derivative of I^L with respect to y:

$$I_y(x,y) = \frac{I^L(x,y+1) - I^L(x,y-1)}{2} \tag{4.5}$$

Spatial gradient matrix:

$$G = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{bmatrix} \tag{4.6}$$

Initialization of iterative L-K:

$$\bar{v}^0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T \tag{4.7}$$

for k = 1 to K with step of 1 (or until $\left\|\overline{n^k}\right\|$ < accuracy threshold)

Image difference:

$$\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + v_x^{k-1}, y + g_y^L + v_y^{k-1}) \quad (4.8)$$

Image mismatch vector:

$$\bar{b}_k = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} \delta I_k(x,y) & I_x(x,y) \\ \delta I_k(x,y) & I_y(x,y) \end{bmatrix} \quad (4.9)$$

Optical flow (Lucas-Kanade):

$$\bar{\eta}^k = G^{-1} - \bar{b}_k \quad (4.10)$$

Guess for next iteration:

$$\bar{v}^k = \bar{v}^{k-1} + \bar{\eta}^k \quad (4.11)$$

end of for-loop on k

Final optical flow at level L:

$$d^L = \bar{v}^K \quad (4.12)$$

Guess for next level L-1:

$$g^{L-1} = \begin{bmatrix} g_x^{L-1} & g_y^{L-1} \end{bmatrix} = 2(g^L + d^L) \quad (4.13)$$

end of for-loop on L

Final optical flow vector:

$$d = g^0 + d^0 \quad (4.14)$$

Location of point on J:

$$v = u + d \quad (4.15)$$

As the next step, the system sends control commands to the camera for panning, tilting and zooming operations. The aim is to have the position of the mean of features of the moving object in the center part of the video and not to miss it. The panning and tilting speeds of the camera are set by using a linear equation depending on the location of the mean of the features with respect to the center of the field of view. The camera moves faster when the object is far away from the center and it moves slower when the object is closer to the center of field of

view. The panning and tilting speeds are calculated independently according to the horizontal and vertical spatial distances. If the mean of features is within a limited distance to the center of field of view, the camera stops. The zooming action is also independent of pan and tilt actions. The maximal distances between the features on the object are calculated. Whenever the object moves away from the camera, this distance decreases and the system sends zoom-in commands to the camera. On the contrary, when the object comes closer to the camera, the distance increases and the system sends zoom-out commands to the camera. As a result, the object can be seen easily on the monitor with a rectangle around it and the user will be warned about that object. Then, the system goes to initial state for further observations in the scene. Since the system operates in many conditions, sometimes there exist errors in the operation, however by some operations such as thresholding the size of an object or setting a limit for the size of an object, these errors are reduced.

The presented system initiates and continuously updates the MGM background model when the camera is stationary before tracking starts. During tracking, the camera moves and background modeling is not performed since a faster operation is desired and the model will have many errors with a moving camera. For re-initialization of the foreground regions and the ROI during tracking, the camera stops at an instance and frame difference is applied in order to have a fast response to the moving object as using frame difference is simple and fast. Then tracking is performed until the next re-initialization.

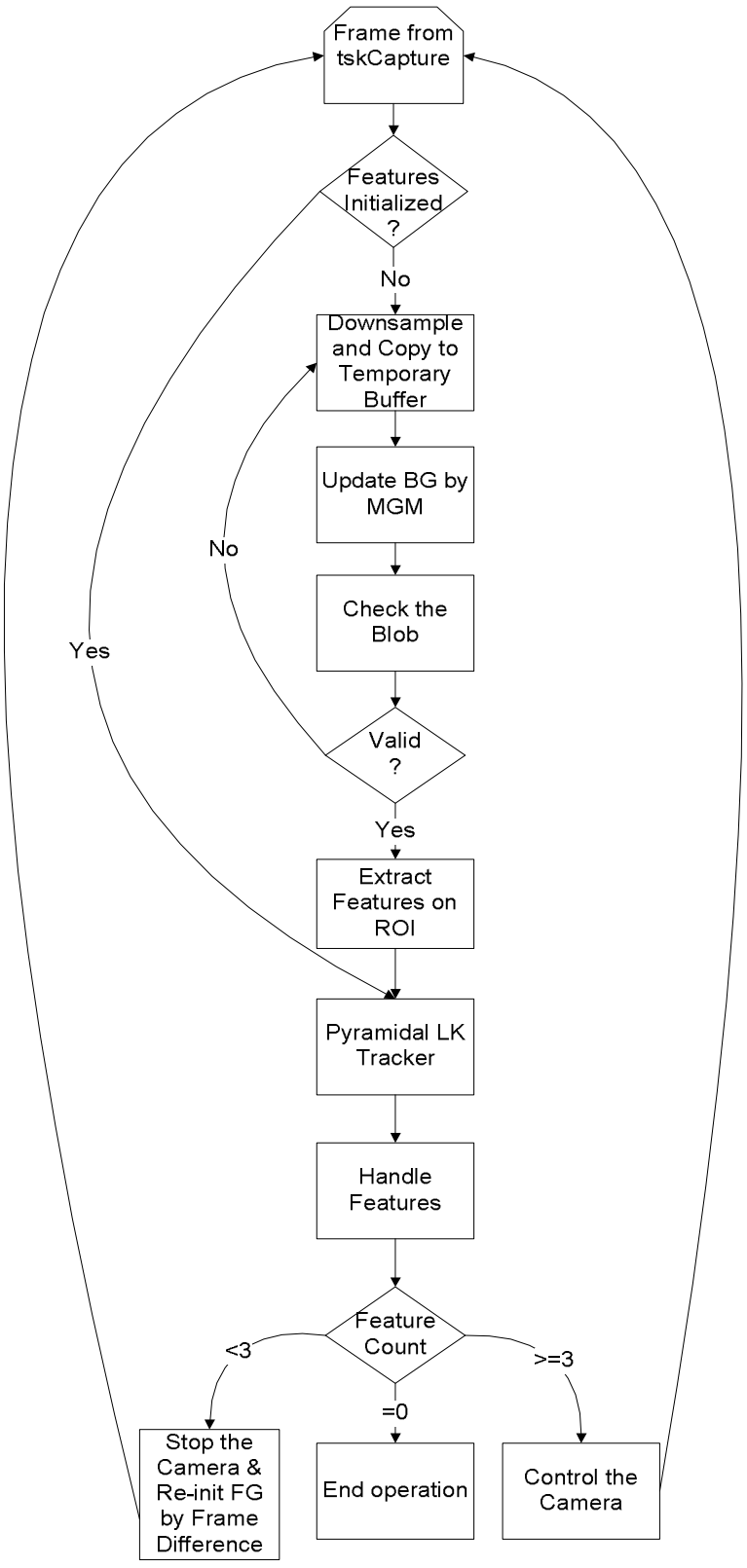


Figure 4.4: Flow Diagram of main body of the code

tskDisplay is responsible for PAL video output. The display channel and video encoder are controlled by this task. The task initializes by setting the video output parameters, allocating variables and buffers, and creating SCOM links for sending and receiving. After initialization, the task continuously checks for processed buffers. When a processed buffer is ready, it is copied to the display buffer by using EDMA and sent to the display channel.

4.3 Testing and Verification

While embedding the code written in C++ programming language to DM642 EVM platform, there exist many challenges. At first, the code needs to be error-free, especially without memory leakage, in order to be run on the real-time system for a long time. Some parts of the code cannot be directly ported to DSP platform since CCS does not support some libraries of Microsoft Visual C++ environment. After those parts are re-written and the code is totally ported to the embedded platform, it is tested by comparing the codes on both platforms. This system is real-time; therefore real-time debugging the code is not easy. Recorded video sequences are used as a reference for debugging. The recorded video sequences are stored in raw data format on the host PC and sent over JTAG connection to DM642 EVM for debugging by using CCS. The same data can be analyzed in Microsoft Visual Studio Environment as well for comparison. However, even with offline sample data the debugging of the whole system is not practical as it has a moving camera but the sample videos are relatively static. Many parts of the code are tested with real-time operation by connecting the full system hardware, generating a test environment and finding the parameters by trials.

CHAPTER 5

PERFORMANCE EVALUATION

5.1 Design of Setup and Test Environment

For some of the applied tests, a remote controlled toy car is used in the laboratory for simulating a real vehicle is moving in a park area. The camera is located on the desk in a position to resemble a security camera which is mounted on a high altitude. The scenarios are applied easily by controlling the toy car.

For the test of real-time operation on outdoor scenes, the system is located on two different locations to detect people or vehicles and the natural effects that are caused by wind, trees, sun and rain, which cause variations in illumination, are analyzed. In various conditions of illumination level and scene, the output video sequences are stored as they are useful for illustration.

In an indoor scene, the effect of a monitor and florescent lights are observed to see the effect of flickering. For this test, toy car is used in the laboratory while a monitor is visible in the scene and the florescent lights are on.

The effects of parameters in the algorithms are also tested using the toy car setup in the laboratory for selecting the optimal parameters in many conditions. The frames of the recorded video sequence are used by a test code as the captured frames in a way to have a periodic

motion on the scene. Then testing various parameters is accomplished by using the same sequence for each case.

On DM642 EVM, the same algorithms are executed. However, the performance of algorithms is not as good as it is executed on PC. The DM642 EVM has a 720 MHz processor while the used PC has a 1.7 GHz Pentium processor. As the algorithms are run in a single fashion and not in parallel by the DSP, the frame rate is lower. The code on DSP has a 6 FPS frame rate, whereas the PC with a 1.7 GHz processor can run the code with a 12 FPS frame rate. If a dual core DSP is used instead of DM642, the performance will be close to the performance of the code on PC.

The screenshots and the results in various conditions are given in Section 5.2 with their explanations.

5.2 Results in Various Conditions

In the first applied test, two video sequences that are captured in two different illumination levels are used. The effects of critical parameters in both cases are analyzed. The first video has a scene with high illumination level (See Figure 5.1).

(a) Frame 10 (b) Frame 70

(c) Frame 90 (d) Frame 155

(e) Frame 250 (f) Frame 305

Figure 5.1: The screenshots of scene at high illumination level

In Figure 5.1, the screenshots taken at some critical times of the scene are illustrated. At Frame 10, there is no object in the scene. At Frame 70, the object starts to enter the scene. At Frame 90, the object is

totally in the scene. Finally at Frame 305, the object completes its action on the scene. While testing the parameters of algorithms, a test code is written. The code starts processing frames from the first frame until the 90^{th} frame. At this time, the blob is detected as the object, and tracking starts with extracting features on the object. The video sequence between the 90^{th} and 305^{th} frames is processed in a way to have an infinite loop of images by playing forward and reverse in this period.

Similarly, in Figure 5.2, the screenshots of the scene which are taken at some critical times are given. There is no object in the scene in 10^{th} frame. In 165^{th} frame, the object starts to enter the scene. The object is totally in the scene at 180^{th} frame and it moves on the scene until the 570^{th} frame. The same written code is applied for this case, too. At start, the code processes frames from the 0^{th} frame until the 180^{th} frame. The blob is detected as the object and the extracted features on this blob are tracked. In order to have a periodic and long-time video sequence, the frames between the 180^{th} and 570^{th} frames are used as input for the algorithms. The algorithm continues to be executed as long as the tracking results are accurate. By applying different parameters of the algorithms, the best performance in both light conditions is tried to be obtained.

(a) Frame 10

(b) Frame 165

(c) Frame 180

(d) Frame 240

(e) Frame 360

(f) Frame 570

Figure 5.2: The screenshots of scene at low illumination level

Totally ten parameters of the algorithms are varied to analyze the effect of them on the performance. These parameters are divided into four groups according to the algorithm they belong to: MGM, Harris Corner Detection, Pyramidal LK Tracking and Delay. The duration of successful tracking is the measure for comparison. These results are given in Table 5.1 and Table 5.2.

Table 5.1: Effects of Parameters on Tracking Duration in High Illumination Level

MGM				**Harris Corner Detection**			**Pyramidal LK Tracking**		**Delay**	**Tracking Duration**
L 1/a	**K**	**T**	**k**	**M**	**min. dist.**	**k**	**win size**	**pyr level**	**input rate**	**# of frames**
200	2	0,7	6	0,01	10	0,040	10,10	3	1	3532 (17)
200	2	0,7	6	0,01	10	0,040	5,5	3	1	1949
200	2	0,7	6	0,01	10	0,040	15,15	3	1	5570
200	2	0,7	6	0,01	10	0,040	10,10	1	1	3318
200	2	0,7	6	0,01	10	0,040	10,10	5	1	4157
200	2	0,7	6	0,01	10	0,040	10,10	3	2	4032
200	2	0,7	6	0,01	10	0,040	10,10	3	3	2435
200	2	0,7	6	0,01	10	0,040	10,10	3	4	343
200	2	0,7	6	0,01	10	0,040	10,10	3	5	80
100	2	0,7	6	0,01	10	0,040	10,10	3	1	3532
500	2	0,7	6	0,01	10	0,040	10,10	3	1	3532
200	2	0,7	6	0,01	10	0,001	10,10	3	1	3591(20)
200	2	0,7	6	0,01	10	0,064	10,10	3	1	4164(16)
200	2	0,7	6	0,01	10	0.200	10,10	3	1	3686(12)
200	2	0,7	6	0,01	10	0.247	10,10	3	1	1709(10)
200	2	0,7	6	0,01	10	0.248	10,10	3	1	567(5)
200	2	0,7	6	0,01	10	0.249	10,10	3	1	213(2)
200	2	0,7	6	0.05	10	0,040	10,10	3	1	2387(12)
200	2	0,7	6	0,6	10	0,040	10,10	3	1	1955(3)
200	2	0,7	6	0,1	10	0,040	10,10	3	1	2387(10)

( ): denotes number of found corners

Table 5.1 (Continued)

200	2	0,7	6	0.001	10	0,040	10,10	3	1	3532(25)
200	2	0,7	6	0.00001	10	0,040	10,10	3	1	3532(31)
200	2	0,7	2.5	0,01	10	0,040	10,10	3	1	0 (wrong FG)
200	2	0,7	0	0,01	10	0,040	10,10	3	1	0 (all FG)
200	2	0.95	6	0,01	10	0,040	10,10	3	1	3532
200	2	0.8	6	0,01	10	0,040	10,10	3	1	3532
200	2	0.6	6	0,01	10	0,040	10,10	3	1	3532
200	1	0,7	6	0,01	10	0,040	10,10	3	1	68
200	3	0,7	6	0,01	10	0,040	10,10	3	1	3532
200	5	0,7	6	0,01	10	0,040	10,10	3	1	3532
200	2	0,7	6	0,01	7	0,040	10,10	3	1	3439(29(
200	2	0,7	6	0,01	20	0,040	10,10	3	1	2387(7)
200	2	0,7	6	0,01	30	0,040	10,10	3	1	2010(3)

( ): denotes number of found corners

Table 5.2: Effects of Parameters on Tracking Duration in Low Illumination Level

MGM				**Harris Corner Detection**			**Pyramidal LK Tracking**		**Delay**	**Result**
L 1/a	**K**	**T**	**k**	**M**	**min. dist.**	**k**	**win. size**	**pyr level**	**input rate**	**# of frames**
200	2	0,7	6	0,01	10	0,040	10,10	3	1	1556(4)
200	2	0,7	6	0,01	10	0,040	3,3	3	1	518
200	2	0,7	6	0,01	10	0,040	5,5	3	1	928
200	2	0,7	6	0,01	10	0,040	15,15	3	1	1727
200	2	0,7	6	0,01	10	0,040	20,20	3	1	2007
200	2	0,7	6	0,01	10	0,040	30,30	3	1	2628
200	2	0,7	6	0,01	10	0,040	10,10	1	1	1371
200	2	0,7	6	0,01	10	0,040	10,10	5	1	1565
200	2	0,7	6	0,01	10	0,040	10,10	3	2	769
200	2	0,7	6	0,01	10	0,040	10,10	3	3	252

( ): denotes number of found corners

Table 5.2 (Continued)

200	2	0,7	6	0,01	10	0,040	10,10	3	4	181
200	2	0,7	6	0,01	10	0,040	10,10	3	5	29
100	2	0,7	6	0,01	10	0,040	10,10	3	1	1567
500	2	0,7	6	0,01	10	0,040	10,10	3	1	1511
200	2	0,7	6	0,01	10	0,001	10,10	3	1	1515(5)
200	2	0,7	6	0,01	10	0,064	10,10	3	1	1515(5)
200	2	0,7	6	0,01	10	0.200	10,10	3	1	1515(5)
200	2	0,7	6	0,01	10	0.247	10,10	3	1	1216(3)
200	2	0,7	6	0,01	10	0.248	10,10	3	1	381(2)
200	2	0,7	6	0.05	10	0,040	10,10	3	1	1507(4)
200	2	0,7	6	0,6	10	0,040	10,10	3	1	1(1)
200	2	0,7	6	0,1	10	0,040	10,10	3	1	1506(3)
200	2	0,7	6	0.001	10	0,040	10,10	3	1	1498(4)
200	2	0,7	6	0.00001	10	0,040	10,10	3	1	1512(4)
200	2	0,7	8	0,01	10	0,040	10,10	3	1	No FG
200	2	0,7	4	0,01	10	0,040	10,10	3	1	1631
200	2	0,7	2.5	0,01	10	0,040	10,10	3	1	wrong FG
200	2	0,7	0	0,01	10	0,040	10,10	3	1	all FG
200	2	0.95	6	0,01	10	0,040	10,10	3	1	1522
200	2	0.8	6	0,01	10	0,040	10,10	3	1	1521
200	2	0.6	6	0,01	10	0,040	10,10	3	1	1522
200	1	0,7	6	0,01	10	0,040	10,10	3	1	767
200	3	0,7	6	0,01	10	0,040	10,10	3	1	1548
200	5	0,7	6	0,01	10	0,040	10,10	3	1	1518
200	2	0,7	6	0,01	1	0,040	10,10	3	1	1280(23)
200	2	0,7	6	0,01	5	0,040	10,10	3	1	1231(10)
200	2	0,7	6	0,01	15	0,040	10,10	3	1	866(3)
200	2	0,7	6	0,01	20	0,040	10,10	3	1	54(2)

( ): denotes number of found corners

L: number of windows in update equations of MGM [8]

K: number of Gaussian distributions

T: the minimum fraction of data to be used for representing the BG
k: number of standard deviations from the mean of the Gaussian
M: corner response function
min.dist.: minimum distance between corners
k: Harris k
win size: size of search window in each pyramidal level
pyr levels: number of pyramidal levels
input rate: a parameter in the code to vary the frame rate

When Table 5.1 and Table 5.2 are analyzed, it can be seen that the size of search window in pyramidal algorithm has a significant effect on the tracking performance. Selecting a small size window leads to lower performance, whereas selecting a bigger sized window leads to higher performance. However, keeping the size of search window will require more calculations, therefore this will reduce frame rate in a real-time system. The frame rate has also an important effect on the performance. With a low frame rate, the displacements of the moving objects in the scene become larger; therefore tracking algorithm may miss these objects or it will require a larger search window to find the objects.

Another parameter, number of pyramidal levels, has also effect on the performance. As the number of pyramidal levels increase, the tracking performance increases. Since the pyramidal structure makes calculations in different resolutions, it enables handling of large pixel motion, therefore the performance becomes better. However, the motion of pixels are limited, therefore too much increasing the pyramidal level number does not further increase performance.

The number of extracted features, corners, is controlled by the parameters of Harris corner detector. If Harris constant, k, corner response function, M, or minimum allowed distance between corners is

decreased, the number of extracted corners raises. The increase in the number of corners, results in an increase in tracking duration up to a certain limit. After that limit, the highest quality features determine the duration of tracking, therefore there will not be a significant increase in the duration.

Due to the effect of Mixture of Gaussians Model on foreground segmentation and feature extraction, the tracking duration changes in the tests. With more number of Gaussian distributions, the background model is more accurate. The performance increases up to K=2 in this test and having more number of Gaussians does not raise the performance. If the scene is more complex, more Gaussians will be more effective for foreground segmentation and may have a more significant effect on detection and tracking performance. In this experiment, it is seen that the performance of scene with low illumination is more sensitive to background modeling, because there is more blur and noise in the low illumination case. When the same parameters are applied to both scenes, it is observed that the segmented foreground regions in low illumination are very smaller in area when compared to the high illumination case. In the experiment, it is observed that by reducing the k parameter, which is the number of standard deviations from the mean of Gaussians, a larger foreground blob is obtained which results in more extracted features and a larger duration of tracking for the low illumination case.

The input rate is the parameter to simulate the various frame rates. When input rate is selected as 1, the original frame rate of the system is observed since all frames are used in processing. When the value is set as 2, the frames are processed in a fashion to use one frame and to skip one frame as the frames are acquired. For input rate of 3, one frame is used in processing and two frames are discarded, therefore the frame rate will be observed as 1/3 of the original rate. It is similar

for case of 4 and 5. The larger value corresponds to a lower frame rate. It can be analyzed that as the frame rate decreases, the tracking performance decreases rapidly.

Figure 5.3: Resistance to flickering effect

In another test (Figure 5.3), the effect of flickering is observed in real-time. In the first and second rows, some frames of input video sequence and the obtained corresponding foreground images are illustrated. It is observed in the figure that the monitor and the florescent lights in the scene do not cause errors in foreground segmentation using MGM. The car is successfully identified as foreground where as the monitor is identified as a background object.

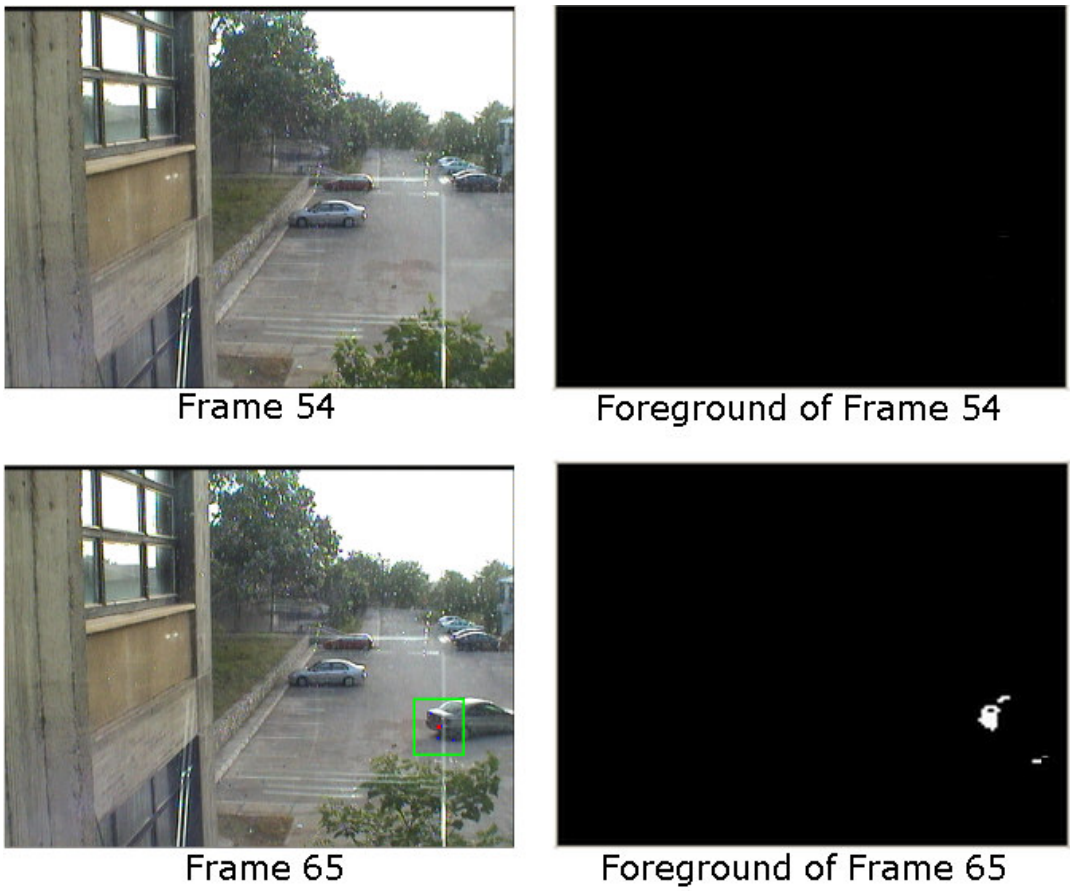


Figure 5.4: The Natural Effects in an Outdoor Scene

In Figure 5.4, the screenshots of real-time operation of the system in an outdoor scene is given. The motion of leaves in the windy air, the raindrops and the motion of trees do not result in an error in background modeling. Only the moving car is identified as the foreground object as desired. MGM, having multimodal character, handles most of the outdoor and indoor scenes effectively.

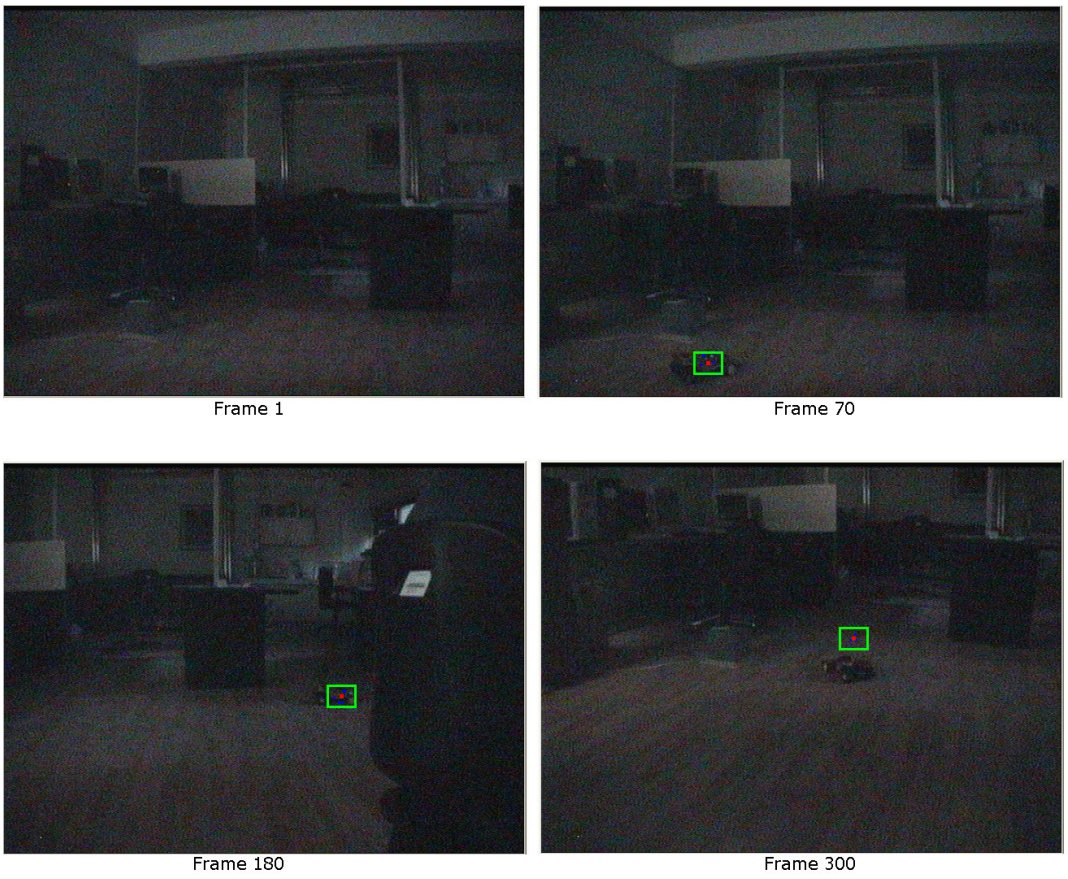


Figure 5.5: Tracking in Low Illumination

Using the same parameters in all scene conditions may not be proper. The screenshots in Figure 5.5 show the tracking ability of the presented system in low light conditions using the parameters used in normal illumination level. The performance is not as good as the illumination case. The tests performed in the laboratory showed that tracking with moving camera is successfully performed for 410 frames in average. Using a camera which works better in low light conditions would give better results.

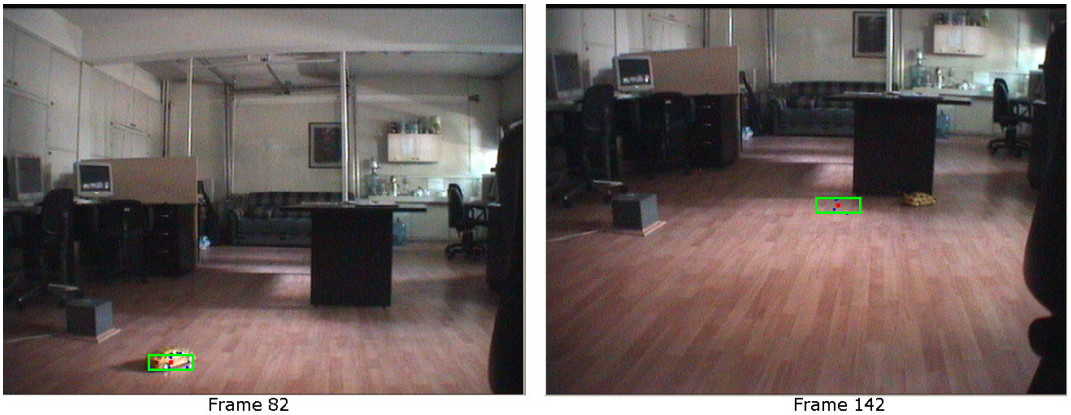


Figure 5.6: An Object with Similar Color of Background

Another weakness of the system is the difficulty of handling an object with a color similar to background. In this case both the background modeling and the tracking algorithms cannot function efficiently. In the given figure, the object is detected in Frame 82; however the tracker misses the object in about 60 frames.

Figure 5.7: Tracking in case of Occlusion

Handling occlusions is another important problem. In Figure 5.7, two people who are walking down the path are occluded while they are walking down the stairs. Some of the features are left behind them.

The algorithm discards the features which are away from the mean of the features. However, in this case most of the features are left behind the people. Therefore, the algorithm cannot handle this case.

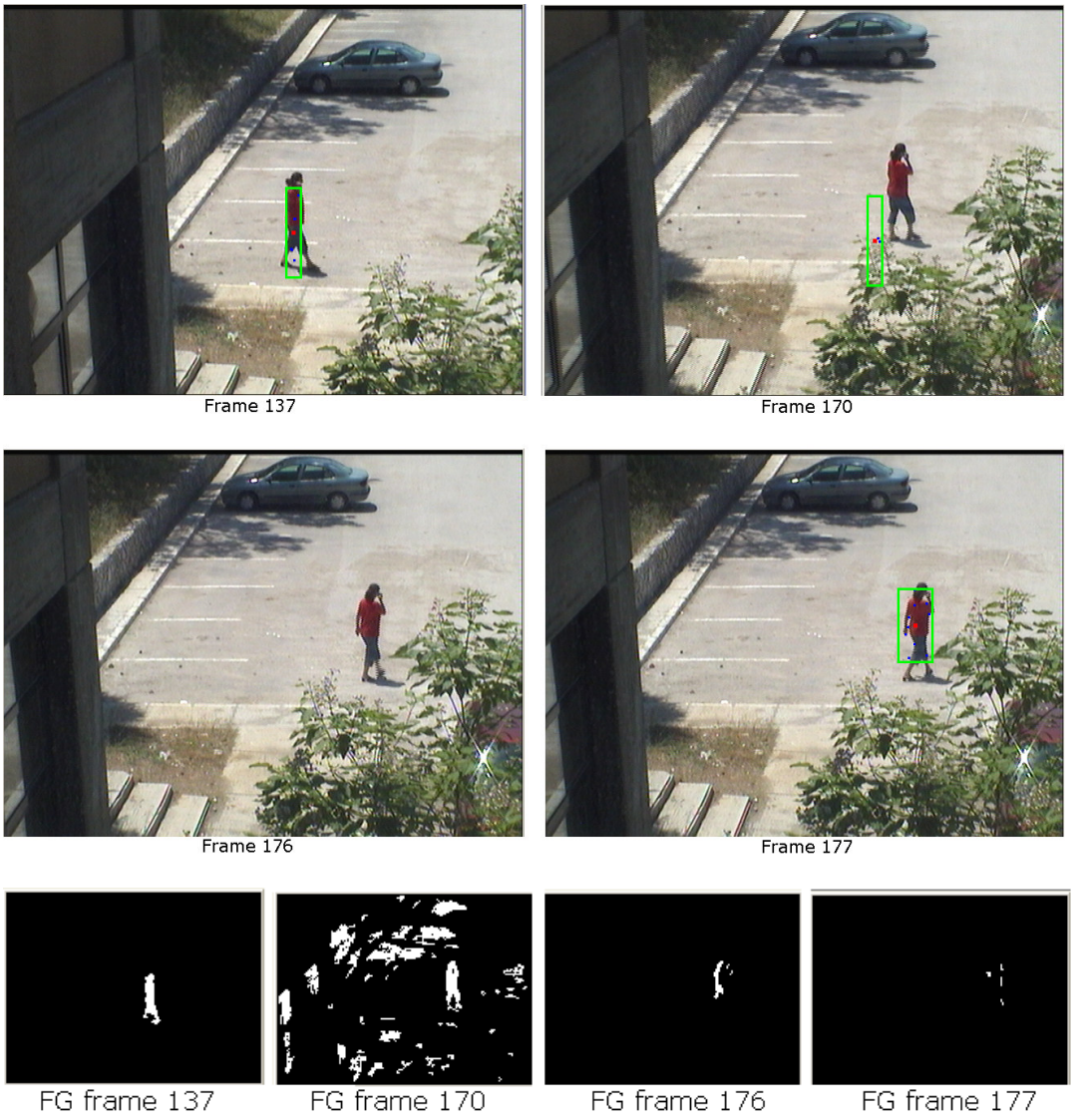


Figure 5.8: Screenshots with their Foreground Images

In the scenario of Figure 5.8, the tracked person is occluded by the tree leaves and since the feature points are lost during occlusion at 170th frame, the tracking stops. However, the system re-initializes the

background and finds the features using the 176th FG image and can continue tracking by the 177th frame. The system re-initializes the features and the model whenever it detects that it missed the object.

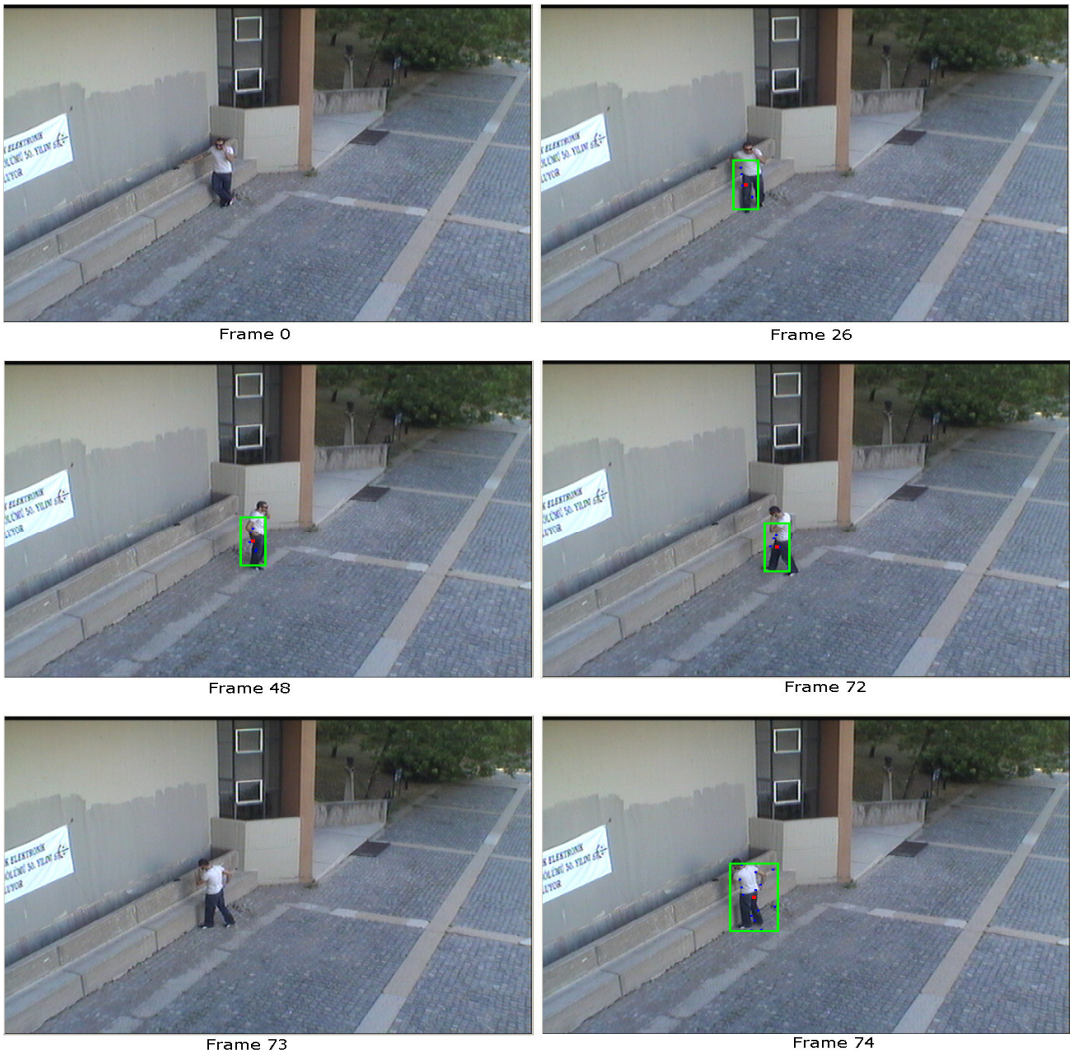


Figure 5.9: Tracking a person when turning around

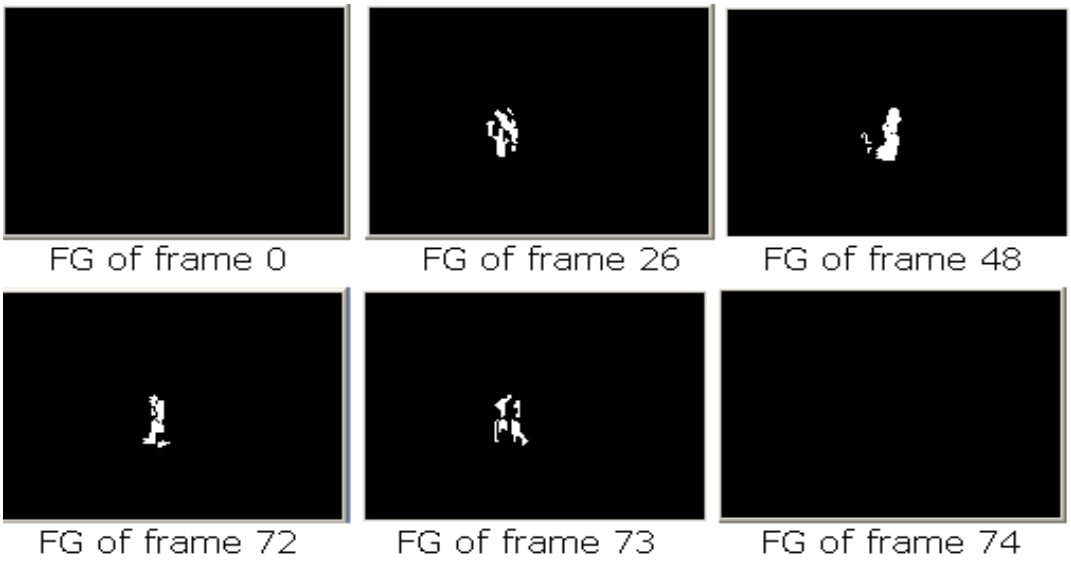


Figure 5.9 (Continued)

In Figure 5.9, the effect of a person turning around himself while waiting is observed. At 26th frame, the person is detected and tracking starts. The person turns around himself in the scene. As the person rotates significantly in Frame 72, the feature points are missed. The system also handles this case by re-initializing the features using the foreground scene information.

The screenshots and the applied tests show that the system works properly in many conditions. With some modifications, the performance may be enhanced further. In complex environments, the number of Gaussians parameter, K, can be increased. Generally, setting K between 3 and 5 performs accurately in many scenes. In case of low illumination, increasing the number of standard deviations from the mean of the Gaussian provides larger foreground blobs and better results. Kalman filtering for tracking the features can be used in the system. The coordinates of each feature in the following frames can be predicted by its dedicated Kalman filter and small occlusions can be handled. However, when the frame rate decreases, the linearity of

motion of features is more distorted because of sudden or complex movements. Thus, the performance of Kalman filter will not be very good.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this thesis, an intelligent system for surveillance applications is presented and implemented. The hardware of the system consists of a Texas Instruments DM642 Digital Video Evaluation Module and a pan-tilt-zoom camera. This system can execute tasks such as video capturing, video processing and event reporting. Processing part involves the automatically detection and tracking of a single moving object using pan-tilt-zoom abilities.

For realizing the presented system, the theoretical bases for background modeling and tracking algorithms, which are necessary for moving object detection, were investigated. Main approaches were discussed by presenting their strengths and weaknesses. By considering its capability of handling various types of scenes, Adaptive Gaussian Mixture Background modeling method was used in the system. The popular and robust Pyramidal Implementation of Lucas-Kanade Feature Tracker was used as the tracking algorithm of the system. The codes developed in Visual Studio Environment were ported to Texas Instruments Code Composer Studio Development Environment for execution on Texas Instruments DM642 DSP. To ensure proper operation, the whole system hardware, software and the ported algorithms were tested by using recorded video sequences, by applying real-time test scenarios or by creating test codes. Since it is difficult to test the whole system in an outdoor place, most of the tests

were performed in the laboratory during development. The results and the screenshots of experiments of the tested system were illustrated.

Developing and executing video processing algorithms on an embedded system with limited processing power and memory is the most critical point in the development of embedded surveillance systems. The presented system in this thesis has a single core DSP of 720 MHz clock and 32 MB external memory. With this configuration, it can perform video processing and loopback with a rate of 6 FPS.

6.2 Future Work

The presented intelligent surveillance system detects only a single object in the scene. As a future work, multiple object detection can be implemented as an additional feature. By properly grouping the features, which is a challenging task, multiple objects can be tracked in the scene. However, tracking multiple objects at the same time will require more processing power.

Another improvement of the system can be accomplished by using the movement information of the camera. As the system can obtain the direction of motion of the moving object and the camera, the background region, which will probably be occupied by the moving object, can be modeled before the object arrives.

Higher level algorithms can be developed on this system in future. After successfully tracking the moving object from one frame to another, the problem of understanding the object behavior follows. The present system can be used for detecting behaviors of objects to identify the events on the scene.

The presented system has many advantages when compared with traditional systems that work on PC. During this thesis work, the

developed algorithms were both tested on PC and the DSP. The performance on PC was better than the performance on DSP because of lower the frame rate on DSP. The system uses a single core DSP to execute the algorithms and a limited memory. As a future work, the algorithms can be run on multiple processors or on a multiple core DSP to have a higher frame rate and hence a higher performance. Furthermore, providing more external memory to the present system will allow more complex algorithms to be developed.

In the presented system, the popular Harris corner detector is used since it is invariant to rotation, changes in illumination and noise. However, this feature detector is not invariant to scaling. Therefore, while using zoom abilities of the camera, some of the features may be lost. By using a feature detector which is insensitive to scaling, the zoom functions of the pan-tilt-zoom camera can be used more efficiently. Thus, in future studies, feature detectors such as Laplacian Harris can be implemented for use in the system.

REFERENCES

[1] Haritaoglu and Flickner, "Detection and Tracking of Shopping Groups in Stores", CVPR 2001.

[2] A. Hampapur et al., "Face Cataloger: Multi-Scale Imaging for Relating Identity to Location", IEEE International Conference on Advanced Video and Signal Based Surveillance, Miami, FL, July 2003.

[3] K. Karmann and A. von Brandt, "Moving object recognition using an adaptive background memory", in Time-Varying Image Processing and Moving Object Recognition II (V. Cappellini, ed.), pp. 289-296, Elsevier, Science, 1990.

[4] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance", in International Conj on Computer Vision, (Kerkyra, Greece), pp. 255-26 1, 1999.

[5] D J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians", in Second IEEE Workshop on Visual Surveillance Fort Collins, Colorado, pp. 74-81. 1999.

[6] Wren, Christopher R., Ali Azarbayejani, Trevor Darrell, and Alex Pentland. "Pfinder: Real-Time Tracking of the Human Body", In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 19, no 7, pp. 780-785.6, July 1997.

[7] Stauffer C, Grimson W. E. L. "Adaptive background mixture models for real-time tracking", in Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149). IEEE Comput. Soc. Part Vol. 2, 1999.

[8] P. Kaewtrakulpong, R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection", in Proceedings of the Second European Workshop on Advanced Video Based Surveillance Systems, Kingston, UK, pp. 149-158, September 2001.

[9] N.M. Oliver, B. Rosario, and A.P. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 831–843, August 2000.

[10] J. Rymel, J. Renno, D. Greenhill, J. Orwell, G.A. Jones, "Adaptive eigen-backgrounds for object detection", Image Processing, 2004. ICIP'04. 2004 International Conference on, vol.3, no.pp. 1847- 1850 Vol. 3, 24-27 October 2004.

[11] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. "Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance", Proceedings of the IEEE, 90(7):1151–1163, 2002.

[12] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking groups of people", Comp. Vis. Image Understanding, vol. 80, no. 1, pp. 42–56, 2000.

[13] A.J. Sindhu and T. Morris, "Resolving Complex Occlusions of Objects during Tracking using Region based Segmentations", in

Proceeding Signal Processing, Pattern Recognition, and Applications, 2006.

[14] Jimin Liang, Yan Chen, Haihong Hu, Heng Zhao, "Appearance-Based Gait Recognition Using Independent Component Analysis", ICNC (1) 2006: 371-380.

[15] D. Magee, "Tracking multiple vehicles using foreground, background and motion models", Image and Vision Computing, 22:143–155, 2004.

[16] S. Messelodi, C. M. Modena, and M. Zanin, "A computer vision system for the detection and classification of vehicles at urban road intersections", Pattern Analysis & Applications, 8, 2005.

[17] C. Stauffer and E. Grimson, "Learning patterns of activity using real-time tracking", IEEE Transactions on Pattern Recognition and Machine Intelligence, 22(8):747–757, August 2000.

[18] H. Veeraraghavan, O. Masoud, and N.P. Papanikolopoulos, "Computer vision algorithms for intersection monitoring", IEEE Transactions on Intelligent Transportation Systems, 4(2):78–89, June 2003.

[19] Hu, W.; Tan, T.; Wang, L.; Maybank, S.; "A survey on visual surveillance of object motion and behaviors", Systems, Man and Cybernetics, Part C, Volume 34, Issue 3, August 2004.

[20] Nicolas Saunier, Tarek Sayed, "A feature-based tracking algorithm for vehicles in intersections", IEEE Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV'06) June 2006.

[21] A. Yilmaz, X. Li, M. Shah, "Contour-Based Object Tracking with Occlusion Handling", PAMI, vol. 26, no. 11, 2004.

[22] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components", IEEE Trans. Pattern Recognition, Machine Intelligence, vol. 23, pp. 349–361, April 2001.

[23] A. Galata, N. Johnson, and D. Hogg, "Learning variable-length Markov models of behavior", Comput. Vis. Image Understanding, vol. 81, no. 3, pp. 398–413, 2001.

[24] Y. Wu and T. S. Huang, "A co-inference approach to robust visual tracking", in Proc. Int. Conf. Computer Vision, vol. II, pp. 26–33, 2001.

[25] M. A. Ali, S. Indupalli, and B. Boufama, "Tracking multiple people for video surveillance", in CRV'06 Workshop on Video Processing for Security (VP4S-06), Quebec City, Quebec, Canada. June 7-9, 2006.

[26] Shijun Sun, David R. Haynor, Yongmin Kim, "VSnakes with local affine deformations", ICIP (2) 2002: 741-744.

[27] S. Fukui, T. Ishikawa, Y. Iwahori, H. Itou, "Extraction of Moving Objects by Estimating Background Brightness", The Journal of The Institute of Image Electronics Engineers of Japan, Vol.33, No.3 pp. 350-357, 2004.

[28] L. Marcenaro, F. Obetti, C.S. Regazzoni, "Change detection methods for automatic scene analysis by using mobile surveillance cameras", in Proc. 2000 International Conference on Image Processing, vol. 1, pp. 244 – 247, 2000.

[29] J. Gao, A. G. Hauptmann, H. D. Wactlar, ”Combining Motion Segmentation with Tracking for Activity Analysis”, The Sixth International Conference on Automatic Face and Gesture Recognition (FGR'04), pp. 699-704, Seoul, Korea, May 17-19, 2004.

[30] Tweed D. and Calway A., “Tracking Many Objects using Subordinated CONDENSATION”, Proc. BMVC 02: 283-292, October 2002.

[31] R. Polana and R. Nelson, “Low level recognition of human motion”, in Proc. IEEE Workshop Motion of Non-Rigid and Articulated Objects, Austin, TX, pp. 77–82, 1994.

[32] B. Coifman, D. Beymer, P.McLauchlan, and J. Malik, “A real-time computer vision system for vehicle tracking and traffic surveillance,” Transportation Res.: Part C, vol. 6, no. 4, pp. 271–288, 1998.

[33] T. J. Fan, G. Medioni, and G. Nevatia, “Recognizing 3-D objects using surface descriptions,” IEEE Trans. Pattern Recognit. Machine Intell., vol. 11, pp. 1140–1157, November 1989.

[34] D.-S. Jang and H.-I. Choi, “Active models for tracking moving objects,” Pattern Recognition”, vol. 33, no. 7, pp. 1135–1146, 2000.

[35] B. D. Lucas and T. Kanade. “An iterative image registration technique with an application to stereo vision”, Proc. International Joint Conference on Artificial Intelligence, pages 674–679, 1981.

[36] C. Tomasi and T. Kanade. “Detection and tracking of feature points”, Carnegie Mellon University Technical Report CMU-CS-91-132, Pittsburgh, PA, 1991.

[37] Jean-Yves-Bouget, "Pyramidal implementation of the Lucas Kanade Feature Tracker", included in Intel Open Source Computer Vision Library (OpenCV) distribution.

[38] Konstantinos G. Derpanis. "The Harris Corner Detector", http://www.cse.yorku.ca/~kosta/CompVis_Notes/harris_detector.pdf, last accessed date: 01 July 2008.

[39] Deependra Talla, Lizy K. John, Viktor Lapinskii, and Brian L. Evans, "Evaluating Signal Processing and Multimedia Applications on SIMD, VLIW and Superscalar Architectures", IEEE 2000.

[40] TI, "The TMS320DM642 Video Port Mini-Driver", Literature Number: SPRA918A, 2003.

[41] Vishal Markandey, Dipa Rao, Texas Instruments, "TMS320DM642 Technical Overview", Application Report Literature Number: SPRU615, September 2002.

[42] Texas Instruments XDS560 Class High Speed Emulators, http://focus.ti.com/docs/toolsw/folders/print/xds560.html, last accessed date: 01 July 2008.

[43] TI, "XDS560 Emulation Technology Brings Real-Time Debugging Visibility to Next-Generation High-Speed Systems", Literature Number: SPRA823A, 2002.

[44] CommFront Communications, http://www.232analyzer.com/RS232_Examples/CCTV/Pelco_D_Pelco_P_Examples_Tutorial.htm, last accessed date: 01 July 2008.