MULTI-CAMERA VIDEO SURVEILLANCE: DETECTION, OCCLUSION HANDLING, TRACKING AND EVENT RECOGNITION

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

BY

OYTUN AKMAN

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

AUGUST 2007

Approval of the Thesis

"MULTI-CAMERA VIDEO SURVEILLANCE: DETECTION, OCCLUSION HANDLING, TRACKING AND EVENT RECOGNITION"

Submitted by **OYTUN AKMAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering** by,

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

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

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

Examining Committee Members

Prof. Dr. Mübeccel Demirekler (*)
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. A. Aydın Alatan (**)
Electrical and Electronics Engineering, METU _____

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Tolga Çiloğlu
Electrical and Electronics Engineering, METU _____

Özkan Özalp (M.S.)
MST İzmir, ASELSAN _____

Date: 16.08.2007

(*) Head of Examining Committee
(**) Supervisor

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 : Oytun Akman

Signature :

ABSTRACT

MULTI-CAMERA VIDEO SURVEILLANCE: DETECTION, OCCLUSION HANDLING, TRACKING AND EVENT RECOGNITION

Akman, Oytun
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. A. Aydın Alatan

August 2007, 153 pages

In this thesis, novel methods for background modeling, tracking, occlusion handling and event recognition via multi-camera configurations are presented. As the initial step, building blocks of typical single camera surveillance systems that are moving object detection, tracking and event recognition, are discussed and various widely accepted methods for these building blocks are tested to asses on their performance. Next, for the multi-camera surveillance systems, background modeling, occlusion handling, tracking and event recognition for two-camera configurations are examined. Various foreground detection methods are discussed and a background modeling algorithm, which is based on multi-variate mixture of Gaussians, is proposed. During occlusion handling studies, a novel method for segmenting the occluded objects is proposed, in which a top-view of the scene, free of occlusions, is generated from multi-view data. The experiments indicate that the occlusion handling algorithm operates successfully on various test data. A novel tracking method by using multi-camera configurations is also proposed. The main idea of multi-camera employment is fusing the 2D

information coming from the cameras to obtain a 3D information for better occlusion handling and seamless tracking. The proposed algorithm is tested on different data sets and it shows clear improvement over single camera tracker. Finally, multi-camera trajectories of objects are classified by proposed multi-camera event recogniton method. In this method, concatenated different view trajectories are used to train Gaussian Mixture Hidden Markov Models. The experimental results indicate an improvement for the multi-camera event recognition performance over the event recognition by using single camera.

Key words: multi-camera surveillance, moving object detection, multi-camera tracking, event recognition, multi-camera occlusion handling.

ÖZ

GÜVENLİK AMAÇLI ÇOKLU KAMERA SİSTEMLERİ: SAPTAMA, ÖRTÜŞME ÇÖZME, TAKİP VE OLAY ANALİZİ

Akman, Oytun
Yüksek Lisans, Elektrik-Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. A. Aydın Alatan

Ağustos 2007, 153 sayfa

Bu tezde, arka plan modelleme, örtüşme çözme, hedef takip ve olay analizi için çoklu kamera sistemleri kullanılan yeni metodlar sunulmuştur. İlk olarak, güvenlik amaçlı tekli kamera sistemlerinin temel yapıları olan hareketli nesne bulma, takip ve olay analizi anlatılmıştır ve bu yapılarda kullanılan çeşitli metodlar performanslarını değerlendirmek amacıyla test edilmiştir. Bir sonraki adımda, güvenlik amaçlı çoklu-kamera sistemleri için iki kamera kullanılarak arka-plan modelleme, örtüşme çözme, hedef takip ve olay analizi metodları sunulmuştur. Çeşitli hareketli nesne saptama metodları incelenmiş ve çoklu-rastgele değişkenli Gaussianlar kullanan bir arka-plan modelleme metodu tercih edilmiştir. Örtüşme çözme çalışmalarında, çoklu-bakış kullanarak örtüşmenin oluşmadığı bir görüntünün (üstten görünüş) üretildiği yeni bir metod sunulmuştur. Önerilen bu metod çeşitli test verileri üzerinde denenmiş ve başarılı sonuçlar alınmıştır. Ayrıca, çoklu kamera kullanılan takip algoritması önerilmiştir. Bu algoritmanın ana fikri, örtüşme çözebilmek ve sürekli takip için kameralardan gelen 2-B bilginin birleştirilerek 3-B bilginin elde edilmesine dayanır. Çeşitli deneysel verilerle incelenen bu metod tekli-kamera takip

metodlarına göre daha başarılı sonuçlar vermiştir. Son olarak, farklı kameralardan gelen gezingelerin birleştirtirilerek tek bir gezingenin oluşturulduğu ve bu gezingenin Gizli Markov Modelleri eğitmek için kullanıldığı bir metod ortaya konmuştur. Yapılan deneylerde, birleştirilen gezingelerle eğitilen HMM lerin olay sınıflandırmasında daha başarılı sonuçlar verdiği gözlemlenmiştir.

Anahtar Kelimeler: Güvenlik amaçlı çoklu-kamera sistemleri, hareketli nesne bulma, çoklu-kamera takip, olay analizi, çoklu-kamera örtüşme çözme

To Mom, Dad and Melis

ACKNOWLEDGEMENTS

I would like to express my gratitude and deep appreciation to my supervisor Assoc. Prof. Dr. A. Aydın Alatan for his guidance, positive suggestions and also for the great research environment he had provided.

I would like to also express my thanks for their great friendship and assistance to Ahmet Saraçoğlu, Yoldaş Ataseven and Cevahir Çığla. We were together for two invaluable years and I will surely miss working with them.

I would like to thank my friends in Multimedia Research Group and Evren İmre for such a friendly research environment they had provided. I have learned much from their suggestions and experiences.

Finally, I would like to thank my Mom, Dad and Melis for their love, support and patience over the years. This thesis is dedicated to them.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The field of machine (computer) vision is concerned with problems that involve interfacing computers with their surrounding environment through visual means. One such problem, surveillance, has an objective to monitor a given environment and report the information about the observed activity that is of significant interest. In this respect, video surveillance usually utilizes electro-optical sensors (video cameras) to collect information from the environment. In a typical surveillance system, these video cameras are mounted on fixed positions or on pan-tilt devices and transmit video streams to a certain location, called *monitoring room*. Then, the received video streams are monitored on displays and traced by human operators. However, the human operators might face many issues, while they are monitoring these sensors. One problem is due to the fact that the operator must navigate through the cameras, as the suspicious object moves between the limited field of view of cameras and should not miss any other object while tracking it. Thus, monitoring becomes more and more challenging, as the number of sensors in such a surveillance network increases. Therefore, surveillance systems must be automated to improve the performance and eliminate such operator errors. Ideally, an automated surveillance system should only require the objectives of application, in which real time interpretation and robustness is needed. Then, the challenge is to provide robust and real-time performing surveillance systems in an affordable price.

With the decrease in costs of off-the-shelf hardware for sensing and computing, and the increase in the processor speeds, surveillance systems have became commercially available, and they are now applied to a number of different applications, such as traffic monitoring, airport and bank security, etc. However, machine vision algorithms (especially for single camera) are still severely affected by many shortcomings, like occlusions, shadows, weather conditions, etc. As these costs decrease almost in daily basis, multi-camera networks that utilize 3D information are becoming more available. Although, the use of multiple cameras leads to better handling of these problems, compared to a single camera, unfortunately, multi-camera surveillance is still not the ultimate solution yet.

There are still challenging problems within the surveillance algorithms, such as background modeling, feature extraction, tracking, occlusion handling and event recognition. Moreover, machine vision algorithms are still not robust enough to handle fully automated systems and many research studies on such improvements are still being done.

1.1 Overview of Related Work

The set of challenges outlined above span several domains of research and the majority of relevant work will be reviewed in the upcoming chapters. In this section, only the representative video surveillance *systems* are discussed for better understanding of the fundamental concept.

Haritaoglu et al. [1] propose a real time visual surveillance system, *W4*, for detecting and tracking multiple people and monitoring their activities in an outdoor environment. The system can identify and segment the objects that are carried by people and can track both objects and people separately. Moreover, it can recognize events between people and objects, such as depositing an object,

exchanging bags, or removing an object. Mittal and Davis [41] present a system, *M2 tracker*, that is capable of segmenting, detecting and tracking multiple people in a cluttered scene by using multiple synchronized surveillance cameras located far from each other. The system is fully automatic, and takes decisions about object detection and tracking by the help of evidence collected from many pairs of cameras.

Beymer et al. [18] developed a system to measure the traffic parameters. The proposed video traffic surveillance system extracts the 3D positions and velocities of vehicles and processes the track data to compute local traffic parameters, including vehicle counts per lane, average speeds, lane change frequencies, etc. These parameters, together with track information (time stamp, vehicle type, color, shape, position), are transferred to the transportation management centers at regular intervals.

Carnegie Mellon University (CMU) and the Sarnoff Corporation (Sarnoff) constructed a video surveillance program, denoted as *'Video Surveillance and Monitoring (VSAM)'* [52]. The objective of the program is to develop a cooperative multi-sensor video surveillance system that provides continuous information over given environment. The system provides the capability to detect moving objects, classify them as human or vehicle, keep track of people, vehicles and their interactions, as well as classify these activities.

1.2 Scope of the Thesis

This thesis is devoted to the problem of defining and developing the fundamental building blocks of automated video surveillance systems via single-camera and multi-camera configurations. The fundamental building blocks can be decomposed into two main parts. The first part consists of the blocks that can be used in single camera configurations as well as in multi-camera configurations.

The second part is based on the methods which use the advantages of multi-camera systems and utilization of 3D information.

For the first part, initial problem, which is the extraction of the objects or features that are subject to interest in surveillance application, is discussed and background subtraction and feature extraction methods are compared. Furthermore, the second problem, which is tracking of extracted features to obtain their motion history, is also discussed and some tracking algorithms are examined. Finally, event recognition which is the higher-level task needed to interpret the motion histories, is explained and a Hidden Markov Model based approach is discussed.

For the second part, background modeling problem is discussed in a multi-camera point of view. A major problem for surveillance applications, which is occlusion, is discussed and a novel algorithm to handle occlusions by using two cameras is proposed. Finally, multi-camera tracking and event recognition by using multi-view data are explained.

1.3 Outline of the Thesis

In Chapter 2, building blocks of single camera surveillance, which are moving object detection, tracking and event recognition, are discussed. Moving object detection algorithms include frame differencing, eigenbackground subtraction, Parzen window based moving object detection and mixture of Gaussians (MOG) based moving object detection, while tracking algorithms include object association, mean-shift tracker, cam-shift tracker and Kanade-Lucas-Tomasi Tracker. In this chapter, the performances of all these algorithms are compared and also simulation results are presented. HMMs are briefly introduced and event recognition by using HMMs is explained.

In Chapter 3, some background material is given about the camera models, epipolar geometry, trifocal tensor and graph theory. These fundamental concepts are utilized in the subsequent chapters of this thesis.

Chapter 4 discusses the multi-camera foreground detection algorithms. Two methods for foreground detection are explained and an algorithm is proposed to model background by using MOG. The performances are compared and some simulation results are presented.

In Chapter 5, an algorithm is presented to handle occlusions by using two cameras. The proposed method starts with the oversegmentation of foreground regions. Next, point transfer by using trifocal tensor is performed and transferred points are grouped together. At the end of this chapter, the simulation results for these methods are given.

Tracking and event recognition algorithms in multi-camera networks are discussed in Chapter 6. A multi-camera tracking method which fuses the information coming from two cameras is proposed. Also, 3D (top-view) locations of the tracked objects are extracted for better visualization of the motion trajectories. A HMM based event recognition method, in which the trajectories of objects in different views are concatenated and used to represent the motion, is explained.

Finally, the summary of the thesis is given in Chapter 7 and the future work plan is suggested.

CHAPTER 2

SINGLE CAMERA SURVEILLANCE

A typical automated single camera surveillance system usually consists of 3 main parts, which can be listed as moving object detection, object tracking and event recognition. In this chapter, some recent studies from the related literature for each main block are discussed, and brief descriptions of several methods are given. Finally, at the end of each part, the examined methods are tested by simulations to analyze their performances.

2.1 Moving Object Detection

The performance of a surveillance system considerably depends on its first step that is detection of the foreground objects which do not belong to the background scene. These foreground regions have a significant role for subsequent actions, such as tracking and event detection.

Moving object segmentation is simply based on a comparison between the input frame and a certain background model, and different regions between the input and the model are labeled as *foreground* based on this comparison. This assessment can be simple frame differencing, if the background is static (has no moving parts and is easier to model).

Figure 2.1 Moving object detection process : Foreground mask is obtained by determining the difference between the input image and the available background model

However, more complex comparison methods are required to segment foreground regions when background scenes have dynamic parts, such as moving tree branches and bushes.

In the literature, there are various algorithms, which can cope with these situations, that will be discussed in the following sections. Moreover, noise removal from the foreground mask, connected component labeling and shadow removal are also discussed in this section.

2.1.1 Moving Object Detection Methods

Although there are numerous proposals for the solution of moving object detection problem in surveillance systems, some of these methods are found out to be more promising by the researchers in the field. Haritaoglu et al. [1] use a two-stage method based on excluding moving pixels from background model computation. In the first stage, a pixel-wise temporal median filter is applied to several seconds of video in order to distinguish moving pixels from stationary points. In the second stage, only those stationary pixels are processed to construct the initial background model.

Elgammal et al. [2] use nonparametric kernel density estimation for background modeling. The model uses pixel intensity (color) as the basic feature for modeling the background and with such a formulation, it can handle cases where the background of the scene is cluttered and not completely static, but contains small motions. The kernel-based model is updated continuously and therefore adapts to changes in the scene background.

Grimson and Stauffer [3] model the values of a particular pixel in the background model as a mixture of Gaussians. Based on the persistence and the variance of each of the Gaussians of the mixture, they determine which Gaussians may correspond to background colors. Proposed method is robust to small motions in the background. KaewTraKulPong and Bowden [4], later improve the adaptation speed of this method [3] by using online expectation maximization.

Oliver et al. [5] adaptively built an eigenspace that models the background. In order to reduce the dimensionality of the space, they use the principal component analysis. Consequently, the segments of an image containing a moving object cannot be well described by this eigenspace model, whereas the static portions of the image can be accurately described as a sum of the various eigenbasis vectors.

2.1.1.1 Frame Differencing Method

Frame differencing could be the simplest moving object detection method, which is based on determining the difference between input frame intensities and background model by using pixel per pixel subtraction, as formulated in (2.1).

$$BD(x,y,t) = \left| I(x,y,t) - BM(x,y,t-1) \right|$$
$$FM(x,y,t) = \begin{cases} 1 & BD(x,y,t) > threshold \\ 0 & BD(x,y,t) < threshold \end{cases} \quad (2.1)$$

where $I(x, y, t)$ is the input frame pixel at time *t* at the location (*x*,*y*), and $BM(x, y, t-1)$ is the background model pixel at time *t-1* at location (*x*,*y*).

Apart from separation, background model could also be adapted in time by using the following equation

$$BM(x, y, t) = \alpha I(x, y, t) + (1-\alpha) BM(x, y, t-1) \tag{2.2}$$

where α is the learning rate; i.e. how fast one would like to update the model in time. This simple method has a drawback for modeling dynamic backgrounds in which moving parts in the background exist.

2.1.1.2 Eigenbackground Subtraction

Eigenbackground modeling [5] is based on constructing an eigenspace by using feature vectors (sample images). The dimension of the constructed eigenspace is reduced by using *Principal Component Analysis (PCA)*. By choosing the *k* principal components (*k* eigenvectors with largest eigenvalues), the most representative *k* vectors that capture the major energy variance are selected. Hence, the lower dimensional subspace, which encapsulates the most common information between the training frames, is constructed. Since background regions are covered in all training frames, such a subspace represents the background.

Next, the incoming frame is projected onto the subspace and reconstructed. The regions that can be reconstructed accurate enough (difference between the original frame and the reconstructed frame is smaller than a certain threshold) are labeled as background, while others are labeled as foreground [6][7].

The most important advantage of this method is due to its real-time foreground-background segmentation performance. However, it is sensitive to the motions in the dynamic background regions and has a major drawback of computationally expensive update procedure.

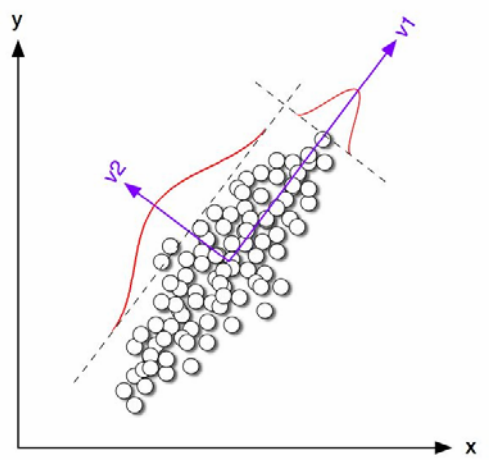


Figure 2.2 Graphical representation of a PCA transformation in two dimensions. The variance of the data in the original Cartesian space (*x*, *y*) is best captured by the basis vectors *v1* and *v2* in a rotated space [19].

2.1.1.3 Parzen Window Based Moving Object Detection

Parzen window based moving object detection method [2] depends on estimating the probability of observing pixel intensity values in a non-parametrical manner, based on the sample intensities. Thus, in the background model, each pixel's probability distribution function is estimated by using Parzen windows for each color channel.

The estimate of the probability of observing a pixel intensity, x, at a particular location on the image is given as

$$p(x) = \frac{1}{N}\sum_{k}\prod_{i=1}^{3}\frac{1}{\sqrt{2\pi\sigma_i^2}}e^{\frac{-(x_i - x_{ki})^2}{2\sigma_i}} \tag{2.3}$$

for three color channels (RGB), by using the assumption that they are all independent.

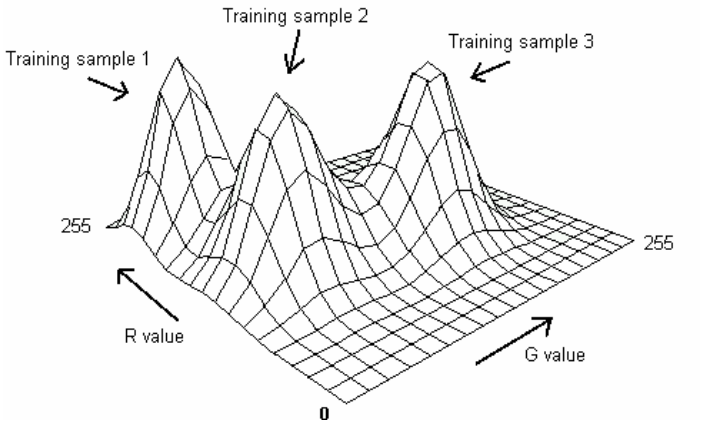


Figure 2.3 Estimated pdf of a pixel by using Red and Green values of training samples.

Suitable kernel bandwidth σ_i must be chosen carefully for each color channel, since a small bandwidth might lead to a ragged density estimate, while relatively wide bandwidths should lead to an over-smoothed density estimate. Thus, for each pixel, differences between its pixel values in consecutive training frames can be found and the mean value of these differences can be used as a kernel bandwidth [2][7]. Then, a new incoming pixel is labeled as background pixel, if the probability of observing that pixel's intensity value in the corresponding

model, as in (2.3), is greater than a certain threshold. Otherwise, it is labeled as foreground pixel. Foreground-background separation by kernel density estimation has an advantage for its robustness in the scenes with cluttered background and small motions.

2.1.1.4 Mixture of Gaussians Based Moving Object Detection

Mixture of Gaussians based moving object detection method [3] depends on modeling of each pixel in the background model by mixture of K Gaussian distributions. Then, the probability of observing pixel value x_N at time N is

$$p(x_N) = \sum_{k=1}^{K} w_k \frac{1}{(2\pi)^{D/2}\left|\sum_k\right|^{1/2}} e^{-\frac{1}{2}(x_N-\mu_k)^T \sum_k^{-1}(x_N-\mu_k)} \tag{2.4}$$

where $\sum_k = \sigma_k^{\,2} I$ (assuming that R,G,B are independent) and w_k is the weight of k^{th} Gaussian.

Every new pixel value, X_t, is compared against the existing K Gaussian distributions, until a *match* is found. A *match* is defined as a pixel value within 2.5 standard deviations of a distribution. If none of the K distributions match the current pixel value, the least probable distribution is replaced with the current X_t by its mean value with an initially high variance, and a low prior weight [3]. The prior weight of the k^{th} Gaussian at time t is adjusted as follows

$$w_{k,t} = (1-\alpha)w_{k,t-1} + \alpha M_{k,t} \tag{2.5}$$

where α is the learning rate and $M_{k,t}$ is '1' for matched models, and '0' for remaining models.

The μ and σ for unmatched distributions remain same while matched ones are updated as follows [3]

$$\begin{aligned} \mu_t &= (1-\varphi)\mu_{t+1} + \varphi x_t \\ {\sigma_t}^2 &= (1-\varphi){\sigma_{t-1}}^2 + \varphi(x_t - \mu_t)^T(x_t - \mu_t) \end{aligned} \tag{2.6}$$

where $\varphi = \alpha\eta(X_t \mid \mu_k, \sigma_k)$.

2.1.2 Simulation Results and Discussion

Discussed methods are tested by using a typical traffic surveillance sequence which is recorded at 25 fps in MPEG-2 format with a resolution of 640x480.

Frame differencing is found out to be very sensitive to small motions happening in the background. As shown in Figure 2.4 (b), branches of the trees and the person waiting by the road are classified as foreground regions. Due to its sensitivity, frame differencing is not suitable for backgrounds with dynamic objects.

Eigenbackground subtraction method is shown to have a better modeling performance compared to frame differencing. However, it is still not adaquate for complex scenes with dynamic background regions.

On the other hand, Parzen window based modeling and MOG based modeling methods have far better results for dynamic scenes. Parzen window based modeling resulted in more noisy foreground regions when compared to MOG based modeling method's foreground masks.

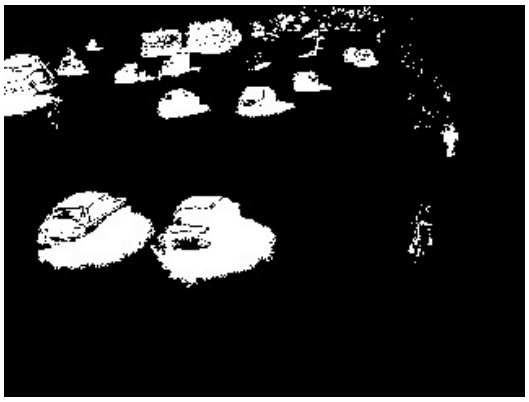
2.1.3 Noise Removal and Connected Component Labeling

Background subtraction algorithms that are discussed in the previous section have quite accurate results. However, it is probable to encounter with noise that deforms the actual shapes of object silhouettes in the foreground masks, and results in false foreground region segmentation. These deformed or false segmented moving objects decrease the performance of object tracking methods and cause inaccurate results. For this purpose, some well-known noise removal methods, called as morphological operators, are usually utilized to remove the noise from the foreground mask [8]. After background subtraction, foreground pixels must also be grouped to form moving objects, since pixel level tracking is not appropriate for surveillance systems. Connected Component Labeling (CCL) [8] method is used for this purpose. Additionally, CCL also helps to filter the objects that have areas which are smaller than some certain threshold value.

2.1.3.1 Morphological Operators

Morphological image processing [8] gets its name from the study of shape, and it has collection of techniques for processing images based on mathematical morphology. Morphological operators work mostly on binary images by using a structuring element and a set operator (intersection, union, complement, etc.). In most of the cases, the structuring element is sized 3×3 and has its origin at the center pixel. It is shifted over the image and at each pixel of the image its elements are compared with the set of the underlying pixels. If the two sets of elements match the condition defined by the set operator (*e.g.* if the set of pixels in the structuring element is a subset of the underlying image pixels), the pixel underneath the origin of the structuring element is set to a pre-defined value (0 or 1) [8]. In this section, some popular morphological operators, such as *erosion* and *dilation,* are discussed briefly.

(a)



(b) **(c)**

(d) **(e)**

Figure 2.4 Simulation results of moving object detection algorithms (a) Input frame (b) Frame differencing result (c) Eigen background result (d) Parzen window result (e) Mixture of Gaussians result

2.1.3.1.1 Erosion

Erosion operator affects the boundaries of the foreground regions [8]. Moreover, it removes the individual pixels that do not have enough neighbor pixels and usually uses the following structuring element;

$$se = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

For each pixel having a value "1" in the structuring element, if the corresponding pixels in the image are a foreground pixel, then the input pixel is not changed. However, if any of the surrounding pixels (considering 4-connectedness) belongs to the background the input pixel is also set to the background value.

(a) **(b)**

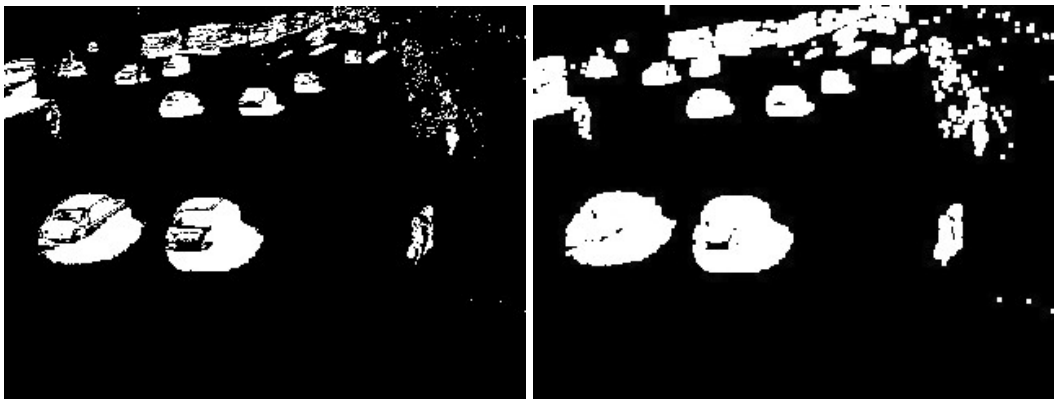
Figure 2.5 Simulation result of erosion operation (a) Input mask (b) Eroded mask

2.1.3.1.2 Dilation

Dilation is the dual operation of erosion [8]. It mainly uses the following structuring element

$$se = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

For each foreground pixel in the image, its 8 neighbor pixels are also set to the foreground value.



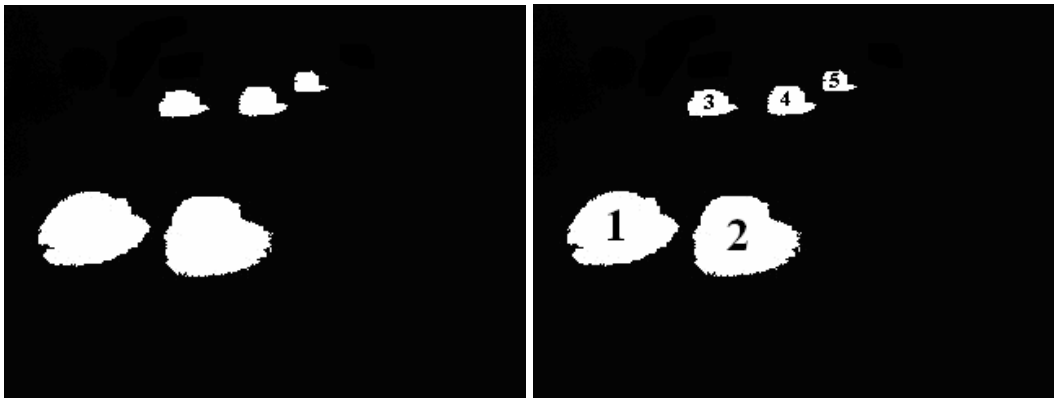
(a) **(b)**

Figure 2.6 Simulation result of dilation operation (a) Input mask (b) Dilated mask

2.1.3.2 Connected Component Labeling

A connected component labeling (CCL) algorithm finds all connected components in an image and assigns a unique label to all points in the same component [8]. Although there are different versions of CCL exists, one of the most popular CCL algorithm is as follows;

1. Scan the binary image to find an unlabeled foreground pixel and assign it a new label L
2. Recursively assign a label L to all its foreground neighbors (having value '1')
3. Stop if there are no more unlabeled foreground pixels
4. Go to initial step



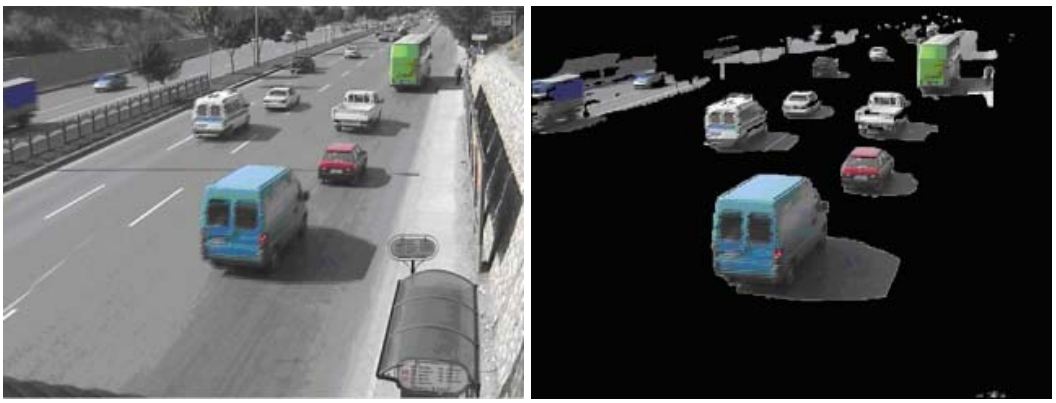
(a) **(b)**

Figure 2.7 Simulation result of ccl (a) Input mask (b) Labeled output mask

2.1.4 Shadow Removal

Moving shadows in the scene might cause false segmentation results in the silhouettes of the moving objects. Additionally, large shadows result in occlusions in the foreground regions. In order to understand shadow removal algorithms, it is useful to identify shadows. Shadows are composed of two parts, *umbra* and *penumbra* [9]. The umbra corresponds to the area where the direct light is totally blocked by the object, whereas light is partially blocked in the penumbra area.

In literature, there are several methods to find and remove shadows [10]. For instance, Mikic et al. [11] propose a statistical parametric approach which is based on the detection of umbra of moving cast shadow in outdoor traffic video scene. Some other approaches prefer to use the HSV color space, since a shadow cast on a background does not change its hue significantly, while its value (V) and saturation (S) values decrease [12].



(a) **(b)**

Figure 2.8 Effects of shadow (a) Input mask (b) Deformed foreground mask

In this thesis, a similar method to [12] is used to remove the shadows. First, foreground regions are obtained by using proper background subtraction algorithm. Then, the shadow detection algorithm in [12] is used to find the shadow pixels in the foreground mask. The algorithm is based on the assumption that the shadow pixels have the same color with the background whereas are of lower brightness. HSV color space is preferred, since hue represents the color type, while saturation and value represent the intensity and brightness. Then, for each foreground pixel, its corresponding pixels in the input frame and background model are compared in HSV color space and shadow mask is determined by using the following relation:

$$SM = \begin{cases} 1 & if \quad \begin{array}{l} |I_H(x,y) - B_H(x,y)| < \tau_H \ \wedge \\ 0 < (B_S(x,y) - I_S(x,y)) < \tau_S \ \wedge \\ 0 < (B_V(x,y) - I_V(x,y)) < \tau_V \end{array} \\ 0 & otherwise \end{cases} \tag{2.7}$$

where $I(x; y)$ and $B(x; y)$ are the pixel HSV values at coordinate (x,y) in the input image and in the background model, respectively.

(a) **(b)**

Figure 2.9 Simulation result of the shadow removal algorithm (a) Input image (b) Shadow removed mask

2.2 Object Tracking

Object tracking is one of the most crucial parts of a surveillance system. After moving object detection is performed, the detected moving objects in consecutive frames must be associated to obtain their trajectories. These trajectories are necessary for interpretation of the behaviors of the moving objects.

2.2.1 Object Tracking Methods

In their seminal work, Tomasi and Kanade [16] proposed a method for tracking some "good" features, that are basically the intensity corners in the image. Their method is based on finding optical flow of the features between two consecutive frames.

Later, Bouguet [17] proposed a pyramidal version of the Kanade-Lucas-Tomasi Tracker. In his method, input frames are downsampled for k times and for each one KLT procedures are repeated, and calculated optical flow is used as initial estimate for next iteration.

Comaniciu et al. [13][14] propose a tracking algorithm of non-rigid objects based on visual features, such as color and/or texture, whose statistical distributions characterize the object of interest. The mean-shift iterations are employed to find the target candidate, which is the most similar to a given target model, with the similarity being expressed by a metric based on the *Bhattacharyya coefficient* [13].

Bradski [15] designed a face tracker where the mean-shift algorithm operates on probability distributions, usually denoted as *cam-shift* in vision literature. In order to track colored objects in video frame sequences, the color image data are represented with a probability distribution by using color histograms.

Beymer et al. [18] use common motion constraint to group features tracked by Kalman filter based tracker. Point features that are found to be moving rigidly together will be grouped together into a single object.

2.2.1.1 Object Association

Object association is one of the simplest tracking approaches. It is based on matching the objects in consecutive frames by using their bounding box locations and color models [7].

Object-i at time t and object-j at time t+1 are matched, $O_i(t) = O_j(t+1)$, if thebounding boxes of i, *Bi* and j, *Bj* are overlapping and, $D(O_i(t), O_j(t+1))$ is smaller than a certain threshold, where *D*(.) is a distance metric between the color histograms of objects. This distance measure could be selected as

Kullback-Leibler divergence [75]

$$D(h_1, h_2) = \sum h_1 \log\left(\frac{h_1}{h_2}\right) + \sum h_2 \log\left(\frac{h_2}{h_1}\right) \tag{2.8}$$

or Bhattacharya coefficient [13]

$$D(h_1, h_2) = \sqrt{1 - \sum \sqrt{h_1 h_2}} \tag{2.9}$$

2.2.1.2 Mean-Shift Tracker

Mean-shift tracker [13][14] is a color model tracker which is based on minimization of a cost function between the target model and candidate models by using mean-shift procedure. Models of the target and candidate objects are constructed by using color histograms and similarity function *d*(*y*) between the target model *q* and the candidate model *p*(*y*) is as follows;

$$d(y) = \sqrt{1 - \underbrace{\sum_{u=1}^{m} \sqrt{p_u(y) q_u}}_{Bhattacharyya\_coeff.}} \tag{2.10}$$

where p and q are m-bin color histograms. By minimizing this function, Bhattacharyya coefficient is maximized. In order to find the mode (local maximum) of the density function (Bhatt. coeff.), the mean-shift vector which is the estimate of gradient vector's density function should be traced. When mean-shift vector converges to a certain region y_0, which maximizes the density function, y_0 becomes the new location of the target object in the next frame.

2.2.1.3 Cam-Shift Tracker

Cam-shift tracker [15] is implemented to handle dynamically changing distributions. For finding the location of the target object in the current frame

1. Model (color histogram) of the target is created
2. Incoming video frame is converted to a probability of the model image, denoted as *backprojection image* [15], where image pixels, having similar intensity values with color model, have larger weights,
3. The mode of probability distribution image around the previous target location is obtained by using mean-shift algorithm
4. The determined mode is the new location of target

2.2.1.4 Kanade-Lucas-Tomasi Tracker

Kanade-Lucas-Tomasi Tracker [16] tracks the feature points which are selected in the video image from one frame to the next frame by means of a gradient method. The features are obtained by using Harris corner detector [24] which is explained in detail in Chapter 5. Displacement of a feature point between two

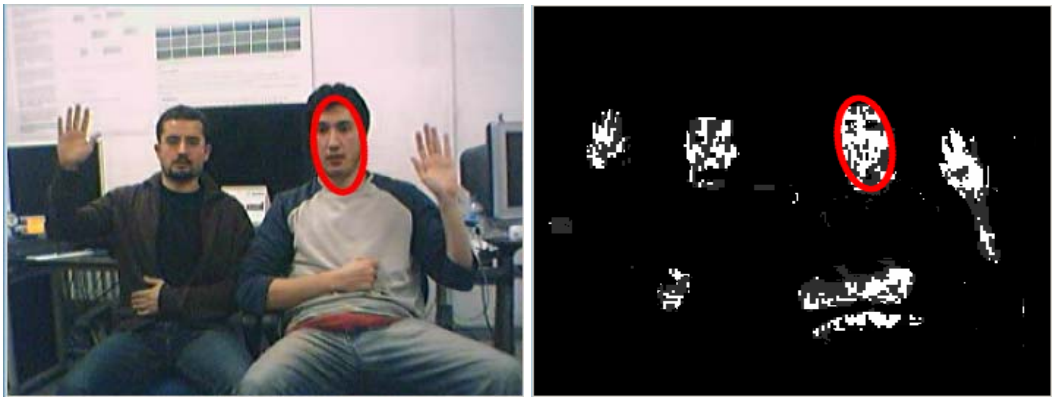
consecutive frames, which is called optical flow vector of that feature, *d*=[*dx dy*] of the feature point (corner) *u*=[*ux uy*] in the first image *I*(*x*,*y*) is found by minimizing the error function ε where;

$$\varepsilon(d)=\varepsilon(d_x,d_y)=\sum_{x=u_x-w_x}^{u_x+w_x}\sum_{y=u_y-w_y}^{u_y+w_y}\left(I(x,y)-J(x+d_x,y+d_y)\right)^2 \qquad (2.11)$$

such that

$$v=[v_x \quad v_y]=[u_x+d_x \quad u_y+d_y] \qquad (2.12)$$

where *v* is new location of *u* in the second image $J(x,y)$, and $I(u)$ and $J(v)$ are similar.



(a) **(b)**

Figure 2.10 Probability image generation (a) Input image (b) backprojection image where the face is used as the color model

In the pyramidal version [17] of the Kanade-Lucas-Tomasi Tracker, performance is improved by downsampled images. In this method, optical flow of the selected

feature point is determined in the k-times downsampled image and it is used as an initial estimate for the same KLT procedure in the next (k-1) times downsampled image. These iterations are achieved until original sized image is reached. Moreover, similar point features are grouped to find the moving objects [18]. Since the trajectories of the moving objects are needed to interpret their motion, the feature points with similar optical flow vectors and close distances are grouped together and the mean of their trajectories is assigned to the corresponding object trajectory.

2.2.1.5 Kalman Tracker

Kalman tracker predicts the new position of the moving object by using a Kalman filter [20][21] with a defined motion model, like motion with constant speed or motion with constant acceleration. A brief discussion about Kalman filter can be found in Chapter 6 to better understand the tracker.

In this chapter, Kalman filter is used as a dynamical system model that supports the main tracker. In each of the tested trackers, the location of the moving object in the next frame is predicted by using a Kalman filter, and then the search for the exact location of the object in the next frame is started from that predicted point. When the exact location is determined, then this location is used to update the filter. Moreover, in 'no measurement case' where the tracker cannot find the object, Kalman filter prediction is accepted as the valid location.

2.2.2 Simulation Results

The aforementioned methods are tested by using various traffic video sequences, which are in MPEG-2 format that are recorded at 25 fps with a resolution of 320x240.

Object association is found to be inappropriate for the surveillance systems, which have severe occlusions, since its basic idea is too simple for complex scenes.

Mean-shift tracker, which is accepted as one of the state-of-the-art tracker, follows the defined color model successfully. The tracking performance of mean-shift algorithm in occluded scenes is very high, if the model for target object is initially well defined. However, in the crowded scenes, such as stop-and-go traffic scenes, where the objects enter the field of view region by occluding each other, there may be the cases that representative model for object is not obtained due to occlusions. Apart from this fact, this algorithm also demands some high computational load and therefore, the real-time performance for multi-object tracking is not suitable for a surveillance system.

KLT tracker is observed to be more robust to the occlusions, since it tracks the features instead of the objects. Moreover, it could be the only solution for the cases in which the application is tested during night in the darkness with conventional RGB cameras, where the background and color models of targets are not available.

However, segmenting the features and grouping them into individual objects in complex scenes is quite difficult to achieve. The features are detected in the previously defined entrance regions and grouped together when they enter the exit regions.

Cam-shift tracker has the same initial model generation problem with mean-shift tracker, however it has a low computational load and it can achieve real time performance in the multi-object tracking.

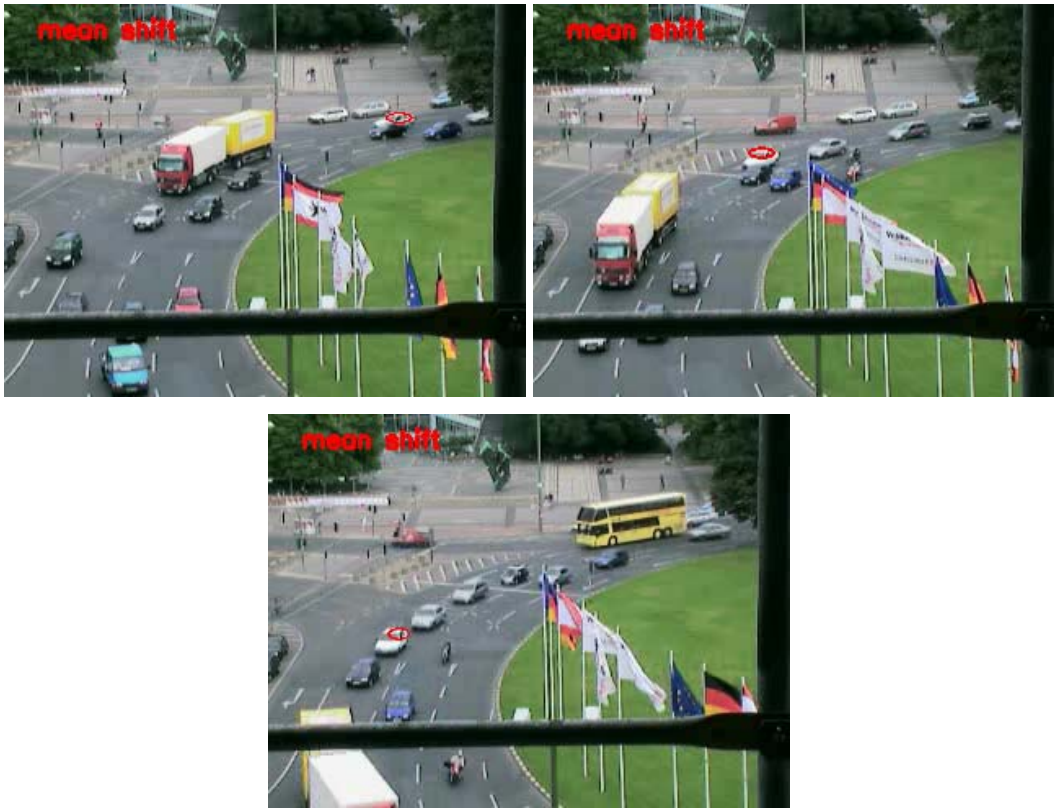
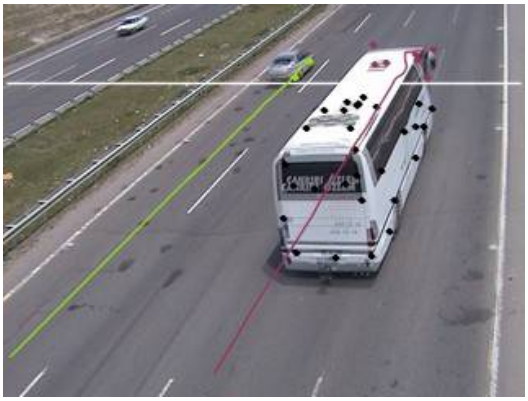


Figure 2.11 Mean-shift tracker for different time instants

Figure 2.12 Cam-shift tracker performance for two moving vehicles whose trajectories are shown



(a)

(b) **(c)**

Figure 2.13 KLT based vehicler tracking results (a) Day scene, (b) and (c) night scene

2.3 Event Recognition

Event recognition is the final part of a typical surveillance system, where the extracted motions of detected objects are interpreted. As the studies in the literature point out, there is no universally acceptable activity type that is of significant interest [7]. They are strongly application dependent. However, a meaningful method to recognize events is separating them as '*normal*' or '*abnormal*' events, where abnormal can be defined as an unusual event, which does not have any previous occurrences through the observation interval.

Many methods have attempted to identify the activities [5][67][68][69][70][71]. Bashir et al. [73] propose a technique to recognize the motion trajectories of objects. They use *Principal Component Analysis* (PCA) to segment the trajectories into similar pieces of motions and employ *Gaussian Mixture Models* (GMMs) to recognize the segmented trajctories. Also in [74], they employ Hidden Markov Models (HMMs) instead of GMMs. The hidden states in HMMS are represented by GMs. In [72], Porikli and Li propose a traffic congestion estimation algorithm that employs Gaussian Mixture Hidden Markov Models (GM-HMM). They extract the congestion features in the compressed domain.

These two methods, [72] and [74], have quite acceptable results and they are pursued as the event recognition algorithm. In this thesis, Gaussian Mixture Hidden Markov Models [22][23] are used to classify trajectories of objects as 'normal' or 'abnormal', after they are trained by trajectories that are accepted to be 'normal'.

2.3.1 Hidden Markov Models

Hidden Markov Model [22] is a statistical method to characterize the spectral properties of the frames of a pattern, which is also called Markov sources or probabilistic functions of Markov Chains.

In this section, some background information about Markov processes is given for better understanding of the HMMs, and HMMs are described with some extensions to Markov processes. Apart this brief explanation, basic problems of HMMs are discussed and finally, event recognition by using HMMs is also examined. Most of the definitions about HMMs follow the text in [22][23] and the reader should refer to these resources for further information.

2.3.1.1 Markov Models

General networks, such as those in Figure 2.14, are denoted as *finite state machines,* and when they have associated state transition probabilities a_{ij}, they are called Markov Networks [22]. These networks are strictly causal which means that the probability of being in one state is dependent on only upon predecessor states. However, for the first order discrete time Markov chains, the probabilistic dependence is truncated to just the previous state, as

$$P(q_t = j \mid q_{t-1} = i, q_{t-2} = k, ...) = P(q_t = j \mid q_{t-1} = i) \tag{2.15}$$

The state transition probability a_{ij} is given

$$a_{ij} = P(q_t = j \mid q_{t-1} = i) \tag{2.16}$$

with the following properties $a_{ij} \geq 0$ and $\sum_{j=1}^{N} a_{ij} = 1$ where N is the number of states.

A Markov model is interpreted as ergodic, if every one of the states has a non-zero probability of occurring given some starting state [22].

2.3.1.2 Extensions to HMMs

Until now, it is assumed that each state in the Markov model corresponds to an observable event. However, the model can also emit some (visible) symbol $v(t)$ (observable events). While sophisticated Markov models allow for the emission of continuous functions (e.g., spectra), at this point for the sake of simplicity, only the case, where a discrete symbol is emitted, is discussed.

The model is then, that in any state, $w(t)$, it has a probability of emitting a particular visible state, $v_k(t)$. This probability is denoted as

$$b_{jk} = P(v_k(t) \mid w_j(t)) \tag{2.17}$$

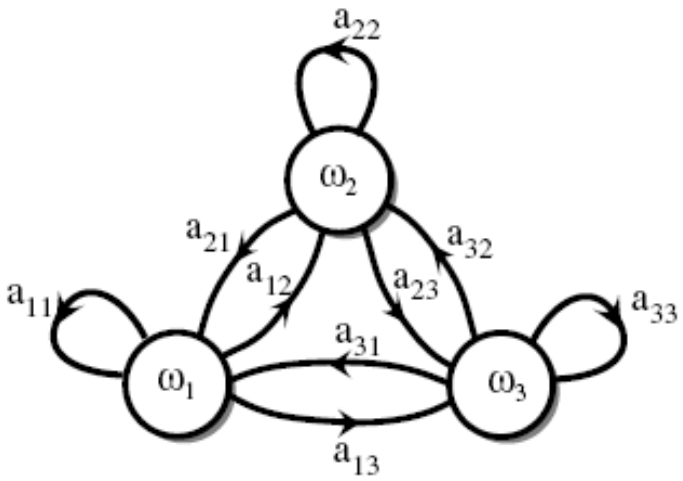


Figure 2.14 A typical finite state machine [22]

Finally, the model is consisted of states, $w(t)$, which have probability of emitting a particular visible state, $v_k(t)$, b_{ij}, with state transition probabilities a_{ij}. If it is accessible only to the visible states, while w_i are not observable, then such a model is called a Hidden Markov Model [22].

The HMM λ can be characterized by five parameters which are

1. Number of states in the model, *N*
2. Number of observation states per state, *M*

3. State transition probabilities, A (a_{ij})
4. Observation probabilities, B (b_{ij})
5. Initial state distributions, π (π_i)

where $\lambda = (A, B, \pi)$ as a compact notation.

2.3.1.3 Fundamental Problems of HMMs

For each HMM, there are 3 critical problems that must be solved. The first one is the evaluation problem that is finding the probability of observation sequence for a given model. The second one is the decoding problem that is finding the optimal state sequence for a given observation sequence, and the last one is the learning problem that is adjusting the parameters of a HMM.

2.3.1.3.1 Evaluation Stage

The probability of observing sequence O for fixed sequence q where $O = (o_1 o_2 ... o_T)$ and $q = (q_1 q_2 ... q_T)$ is

$$P(O \mid q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) \tag{2.18}$$

where it is assumed that observations are statistically independent. Then, the probability of observing sequence O for given model λ is;

$$P(O \mid \lambda) = \sum_{allq} P(O, q \mid \lambda) = \sum_{allq} P(O \mid q, \lambda) P(q \mid \lambda) \tag{2.19}$$

Above probability can be solved inductively in an efficient way by using the forward-backward algorithm [22] as follows;

1 Initialization

$$\alpha_1(i) = \pi_i b_i(o_1) \qquad 1 \le i \le N$$

2 Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}) \qquad \begin{matrix} 1 \le t \le T-1 \\ 1 \le j \le N \end{matrix}$$

3 Termination

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

2.3.1.3.2 Decoding Stage

There are more than one solution to find the 'optimal' state sequence associated with the given observation sequence. The problem lies within the definition of the optimal state sequence, since there are several possible optimality criteria. For example, when the HMM has zero state transition probabilities, the 'optimal' sequence may not be a valid one. In order to find a single best state sequence, *Viterbi Algorithm* [23] can be used

For finding the best state sequence q for the given observation sequence O, it is required to define the quantity as follows

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} (P(q_1 q_2 \ldots q_{t-1}, q_t = i, o_1 o_2 \ldots o_t \mid \lambda)) \qquad (2.20)$$

where $\delta_t(i)$ is the highest probability along a single path at time *t* which accounts for the first *t* observations and ends in state *i*.

By induction

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}].b_j(o_{t+1}) \tag{2.21}$$

The complete procedure for finding the best state sequence can now be stated as follows;

1. Initialization

$$\delta_1(i) = \pi_i b_i(o_1) \qquad 1 \le i \le N$$
$$\psi_1(i) = 0$$

2. Recursion

$$\delta_t(j) = \max_{1 \le i \le N}[\delta_{t-1}(i) a_{ij}] b_j(o_t) \qquad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix}$$

$$\psi_t(j) = \arg\max_{1 \le i \le N}[\delta_{t-1}(i) a_{ij}] \qquad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix}$$

3. Termination

$$P^* = \max_{1 \le i \le N}[\delta_T(i)]$$
$$q_T^* = \arg\max_{1 \le i \le N}[\delta_T(i)]$$

4. Path backtracking

$$q_t^* = \psi_{t+1}(q^*_{t+1}) \qquad t = T-1, T-2, ..., 1$$

2.3.1.3.3 Learning Stage

The most difficult problem of HMMs is to determine the model parameters (A, B, π) to satisfy a certain optimization criterion. There is no defined way to analytically solve for the model parameter set that maximizes the probability of observation sequence. However, $\lambda = (A, B, \pi)$ can be selected such that its

likelihood, $P(O|\lambda)$, is locally maximized by using an iterative procedure, such as the *Baum-Welch Method* [23].

The probability of being in state-i at time *t*, and state-j at time *t*+1, given the model and the observation sequence, $\varepsilon_t(i,j)$ is defined as

$$\varepsilon_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, \lambda) \tag{2.22}$$

By using the definitions of forward and backward variables, $\varepsilon_t(i,j)$ can be written as

$$\varepsilon_t(i,j) = \frac{P(q_t = i, q_{t+1} = j, O \mid \lambda)}{P(O \mid \lambda)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \tag{2.23}$$

Then, the probability of being in state-i at time t, given the entire observation sequence and the model, $\gamma_t(i)$ is related with $\varepsilon_t(i,j)$ as follows

$$\gamma_t(i) = \sum_{j=1}^{N} \varepsilon_t(i,j) \tag{2.24}$$

Then, $\sum_{t=1}^{T-1} \gamma_t(i)$ can be interpreted as expected number of transitions from state-i in *O* while $\sum_{t=1}^{T-1} \varepsilon_t(i,j)$ can be interpreted as expected number of transitions from state-i to state-j in *O*.

By using these formulas, a set of reasonable reestimation formulas for π, A, and B is

$\bar{\pi}_j$ = *expected number of times in state at time t=1* = $\gamma_1(i)$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state i to j}}{\text{expected number of transitions from state i}} = \frac{\sum_{t=1}^{T-1} \varepsilon_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state j and observing } v_k}{\text{expected number of times in state j}} = \frac{\sum_{t=1,\ s.t.\ o_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

It is stated in [23] that, if the current model is defined as $\lambda = (A, B, \pi)$ and the reestimated model is defined as $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$, then either $\lambda = \bar{\lambda}$; or $\bar{\lambda}$ is more likely than λ in the sense that $P(O \mid \bar{\lambda}) = P(O \mid \lambda)$.

If the reestimated model $\bar{\lambda}$ is used in place of the current model λ, and the reestimation calculation are repeated, then a certain limiting point which is the ML estimate of the HMM is reached.

2.3.1.4 Continuous Observation Densities in HMMs

Observations of HMMs are considered as discrete symbols. However, in some applications, the observations are continuous signals and discretization causes loss of information. Hence, HMMs with continuous observation densities [23] are used to model continuous signals.

The most general representation of the observation pdf is a finite mixture of Gaussians;

$$b_j(o) = \sum_{k=1}^{M} c_{jk} N(o; \mu_{jk}, U_{jk}), \quad 1 \le j \le N \tag{2.25}$$

where o is the observation vector modeled and c_{jk} is the weight of k^{th} mixture at state-j.

2.3.2 Event Recognition by Using HMMs

Motion history of a moving object must be extracted to interpret its behavior. The position and the velocity information of the object of interest can be used to define its motion history. Furthermore, the position is sufficient enough to model the motion history when the time between the position measurements is known, because position information together with time information encapsulates the velocity.

Positions of the moving objects, during the observation interval, are obtained by using the previously discussed moving object segmentation and tracking methods in each input frame. The position of the i^{th} object in the k^{th} frame is shown as the position vector;

$$p_k^i = \begin{bmatrix} x_k^i & y_k^i \end{bmatrix} \tag{2.26}$$

Then, these position vectors are concatenated to form a trajectory vector, in order to generate a motion history, which is defined as;

$$tr(i) = \begin{bmatrix} x_m^i & y_m^i \\ x_{m+1}^i & y_{m+1}^i \\ . & . \\ . & . \\ x_n^i & y_n^i \end{bmatrix} \tag{2.27}$$

where *m* is the starting frame number in which the object enters the field-of-view (FOV) region, *n* is the end frame number in which the object leaves the FOV region.

As stated in the previous sections, GM-HMMs should be trained by 'normal' data to classify 'abnormal' data. Therefore, trajectories, which are known to be 'normal', are used to train GM-HMMs.

In order to give some idea, the discussed method is tested by using a traffic video sequence which is in MPEG-2 format recorded at 25 fps with a resolution of 320x240.

Single GM-HMMs is found to have difficulties in modeling the whole traffic flow, since each lane has different characteristics. For example, 30 kmph is a 'normal' speed for the rightmost lane, whereas it is 'abnormal' (too slow) for the leftmost lane in free-flow traffic. Hence, the same number of GM-HMMs as traffic lanes is used to model the traffic flow. Moreover, the number of states in the GM-HMMs of comparatively fast lanes is small with respect to other lanes, since fewer measurements are available for those lanes. These models are illustrated in Figure 2.15.

After GM-HMMs are trained by the observed "normal" sequences, trajectories of the moving objects are classified by GM-HMMs of the corresponding lane and decided to be 'normal' or 'abnormal'. If the given trajectory can be modeled by

the corresponding GM-HMMs, which means that the calculated Viterbi distance [23] is smaller than a certain threshold, then the trajectory is classified as ‘normal’; otherwise, as ‘abnormal’. A classification example is given in the Figure 2.16.

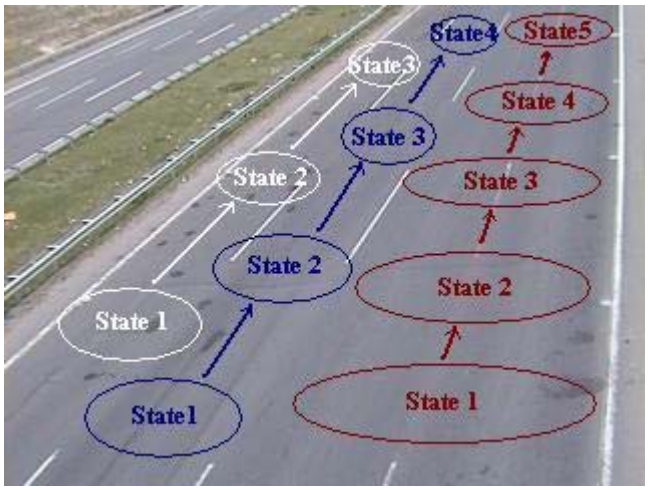
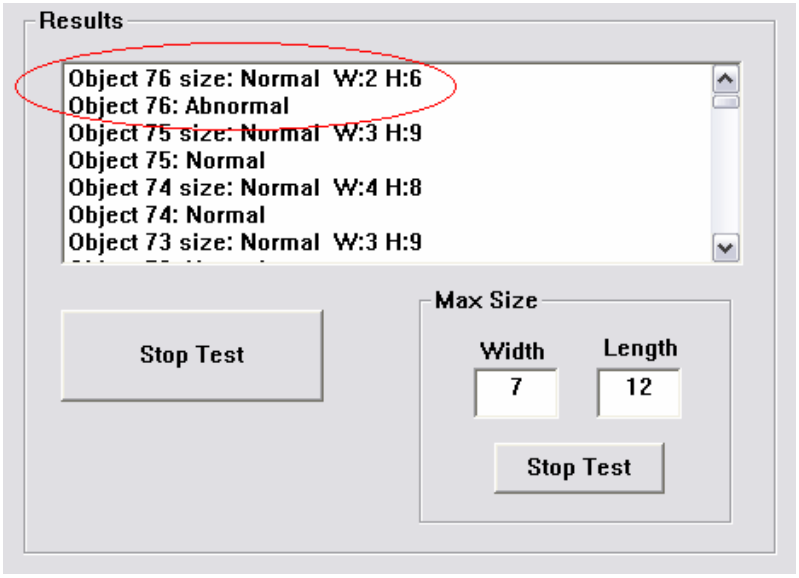


Figure 2.15 HMM lay out example

(a) **(b)**



(c)

Figure 2.16 Event detection by using GM-HMMs (a) object with horizontal motion (b) object leaves the road (c) classified as 'abnormal', since GM-HMMs are trained with vertical trajectories

CHAPTER 3

BACKGROUND MATERIAL

In this chapter, some fundamental concepts, which are strictly necessary for better understanding of the remaining chapters in this thesis, are briefly discussed. This chapter contains information about important material, such as camera model, epipolar geometry, fundamental matrix, trifocal tensor, as well as an introduction to the graph theory. Most of the definitions about camera model, epipolar geometry, fundamental matrix and trifocal tensor follow the text in [24][25] and the reader should refer these resources for further information.

3.1 Camera Model

A camera model is a simple mapping between the 3D world coordinates and a 2D image. Transformation between the 3D world coordinates and a 2D image are mostly represented as a simple matrix. Throughout this thesis, the camera model with finite optical center, which is denoted as finite camera [24], is utilized. In the beginning, the most specialized and simplest finite camera model, which is the *basic pinhole camera*, is explained and then this model is progressively generalized through a series of modifications.

3.1.1 Basic Pinhole Camera Model

In basic pinhole model [24], projection of a point in 3D world coordinates onto 2D image plane or focal plane (or Z = f plane) is simply achieved by passing a line between the 3D point and the camera center. Intersection of this line with the image plane is the location of the projection of the 3D point on the 2D image plane, as shown in the Figure 3.1.

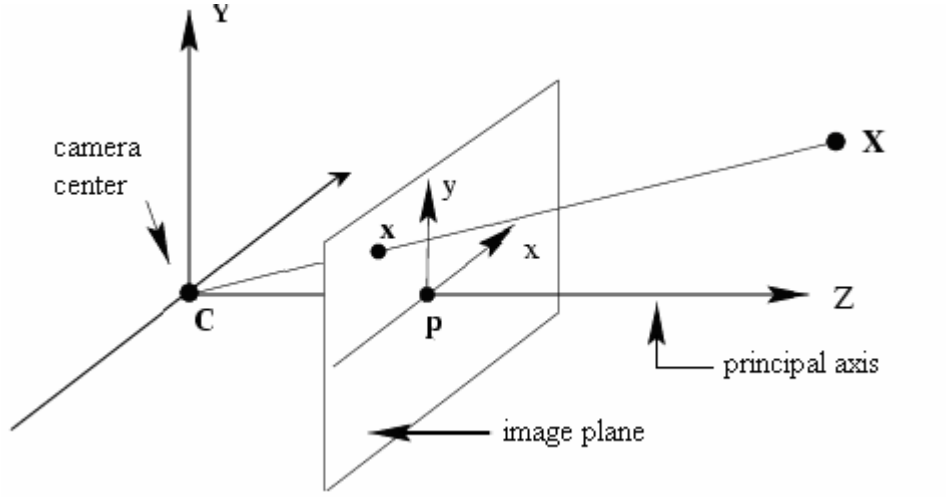


Figure 3.1 Basic pinhole camera model

By using similar triangles as shown in Figure 3.2, it can easily be computed that 3D point $X = (X, Y, Z)^T$ on the space is mapped to the 2D point on the image plane.

$$(X, Y, Z) \mapsto (fX / Z,\ fY / Z) \tag{3.1}$$

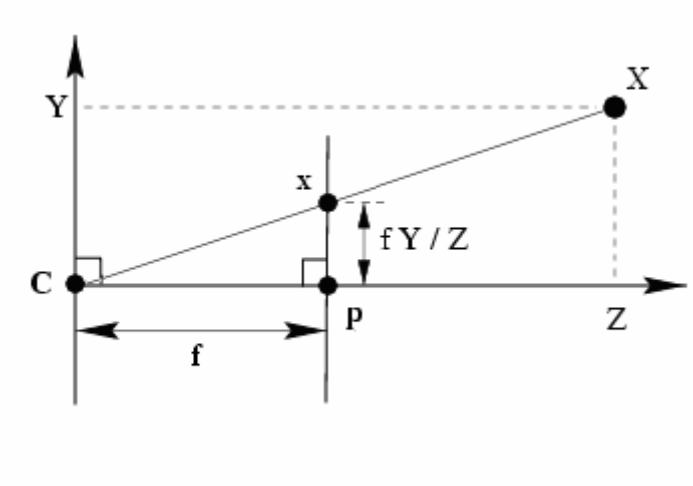


Figure 3.2 Side view of the basic pinhole camera model

If both 3D (world) and 2D (image plane) point coordinates are expressed by using homogeneous vectors, then the projection is a linear mapping between their homogeneous coordinates. Then, (3.1) can be written as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} [I \mid 0] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.2}$$

And (3.2) can be written more compactly as

$$\overline{x} = P\overline{X} \tag{3.3}$$

where $\overline{x} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix}$, $\overline{X} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$ and $P = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix}$.

Equation (3.3) defines the 3x4 camera matrix (camera projection matrix) P for the basic pinhole model of central projection as [24]

$$P = diag(f, f, 1)[I \mid 0] \tag{3.4}$$

3.1.2 Principal Point Offset

Intersection point of the principal axis and image plane is called 'principal point' [24] (P in Figure 3.3). In expression (3.1), it is assumed that the origin of the image plane is at the principal point. However, in practical cases this relation may not hold, and hence, the mapping becomes

$$(X, Y, Z)^T \mapsto (fX / Z + p_x, fY / Z + p_y)^T \tag{3.5}$$

where (p_x, p_y) are the coordinates of the principal point seen on Figure 3.3. This relation can be written in matrix form

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.6}$$

then (3.6) has the concise form

$$\bar{x} = K[I \mid 0]X_{cam} \tag{3.7}$$

where $$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \tag{3.8}$$

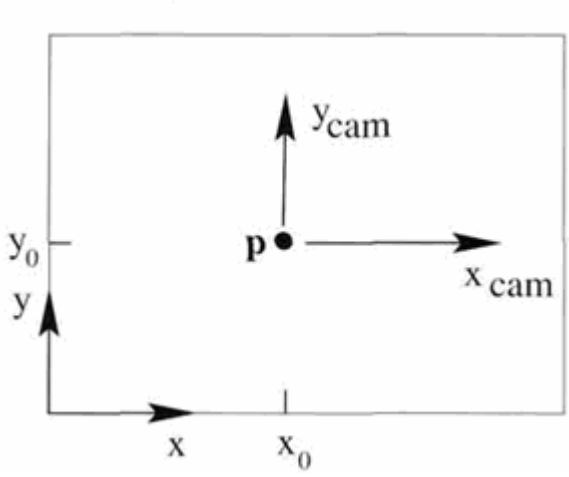


Figure 3.3 Principal point of the image plane

The matrix K is called the *camera calibration matrix* [24] and the camera is assumed to be located at the origin of a Euclidean coordinate system with its principal axis pointing straight down the Z-axis, and the point X_{cam} is expressed in this coordinate system. Such a coordinate system is called the *camera coordinate frame*.

3.1.3 Camera Rotation and Translation

Points in space can be expressed in terms of a different Euclidean coordinate frame which is known as the *world coordinate frame*, and it is related to camera coordinate frame via a rotation and a translation, as shown in Figure 3.4. If X is an inhomogeneous 3-vector representing the coordinates of a point in the world coordinate frame, and X_{cam} represents the same point in the camera coordinate frame, then $X_{cam} = R(X - C)$ and this relation can be written can be written in homogeneous coordinates as

$$X_{cam} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} X \tag{3.9}$$

where C represents the coordinates of the camera centre in the world coordinate frame, and R is a 3 x 3 rotation matrix representing the orientation of the camera coordinate frame.

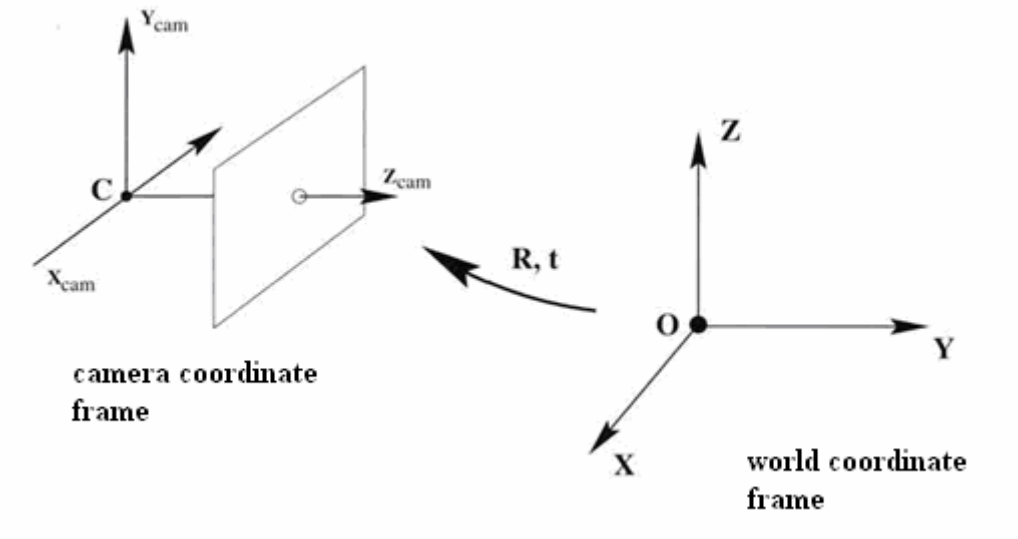


Figure 3.4 Transformation between world coordinate frame and camera coordinate frame

Then (3.7) and (3.9) lead to

$$P = K[R \mid t] \tag{3.10}$$

where $t = -RC$.

3.1.4 Computation of the Camera Matrix P

In computation of the camera matrix *P*, point correspondences $X_i \leftrightarrow x_i$ between 3D points and 2D image points are used. The camera matrix *P* must satisfy the relation

$$\bar{x}_i = P\bar{X}_i \tag{3.11}$$

For all correspondences where $P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix}$, $\bar{x}_i = \begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix}$ and $\bar{X}_i = \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix}$.

Then using (3.11)

$$\begin{bmatrix} 0^T & -w_i {X_i}^T & y_i {X_i}^T \\ w_i {X_i}^T & 0^T & -x_i {X_i}^T \\ -y_i {X_i}^T & x_i {X_i}^T & 0^T \end{bmatrix} \begin{pmatrix} P^1 \\ P^2 \\ P^3 \end{pmatrix} = 0 \tag{3.12}$$

is derived where each P^{iT} is a 4-vector, which is the i^{th} row of P. Since three equations in (3.12) are linearly dependent, then

$$\begin{bmatrix} 0^T & -w_i {X_i}^T & y_i {X_i}^T \\ w_i {X_i}^T & 0^T & -x_i {X_i}^T \end{bmatrix} \begin{pmatrix} P^1 \\ P^2 \\ P^3 \end{pmatrix} = 0 \tag{3.13}$$

The above relation could be used instead of (3.12).

The projection matrix *P* is computed by solving a set of equations $Ap = 0$, where *p* is the vector containing the entries of the matrix *P*. Since the camera matrix has 12 entries and 11 degrees of freedom, 11 equations are needed to solve for *P*. For each point correspondence, one has 2 equations, hence, 6 point correspondence is enough to solve for *P*. Then, the algorithm [24] is

(i) For each correspondence $X_i \leftrightarrow x_i$, compute the matrix A_i (only the first two rows are required in general).

(ii) Assemble the n 2x9 matrices A_i into a single 2nx9 matrix *A*.

(iii) Obtain SVD of *A*. The unit singular vector corresponding to the smallest singular value is the solution *h*. Specifically, if $A = UDV^T$ with *D* diagonal with positive diagonal entries, arranged in descending order down the diagonal, then *h* is the last column of *V*.

(iv) The matrix *P* is determined from *p* as in (3.12).

3.1.5 Homography Matrix

Homography [24] is a projective mapping that maps coplanar points x_i on the plane π_1 to coplanar points x_i' on the plane π_2. Homography can be represented by 3x3 matrix, *H,* which can be derived by using the mapping between the points x_i and x_i'.

The matrix *H* contains 9 entries, but it can be defined only up to a scale. Thus, the total number of degrees of freedom in a 2D projective transformation is 8. Therefore, we need 4 points to compute *H* matrix, since a 2D point has two degrees of freedom corresponding to the *x* and *y* components. Computation procedure is similar to that of the projection matrix *P*.

3.2 Epipolar Geometry and Fundamental Matrix

The epipolar geometry [24] is the intrinsic projective relation between two views. It only depends on the internal parameters and relative positions of the cameras; therefore, it is independent of the scene structure. *Fundamental matrix* [24], *F*, represents the epipolar geometry between two views and encapsulates the intrinsic geometry.

3.2.1 Epipolar Geometry

The epipolar geometry [24] between two views is the geometry of the intersection of two image planes with the line between the two camera centers. The intersection point of the line joining two camera centers (the baseline) with the image plane is denoted as *epipole*. Therefore, the epipole is the image in one view of the camera center of the other view. The plane containing the baseline is called as *epipolar plane* and the line formed by intersecting the epipolar plane with the image plane is called *epipolar line*. All epipolar lines intersect at the epipole and an epipolar plane intersects the image planes in epipolar lines.

In the case where only 2D image coordinates, *x*, of the 3D point *X* are known, epipolar plane can be obtained by using the ray connecting *x* and camera center and baseline. Then, the intersection of this epipolar plane and the other image plane will be the epipolar line and the correspondence of *x* in the second view x' which is the projection of *X* onto second image plane, must be on this epipolar line. By using some stereo matching algorithms, location of the x' in image plane can be determined. Hence, the intersection of the ray passing through the first camera center and x with the ray passing through the second camera center and x' is the 3D location of the X.

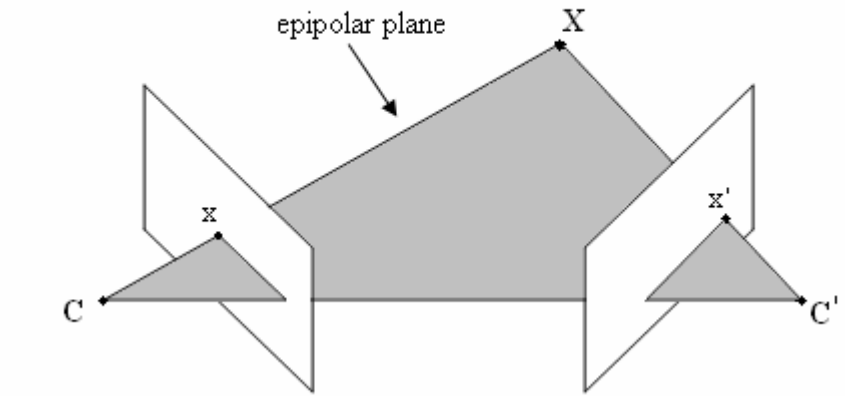


Figure 3.5 Epipolar geometry

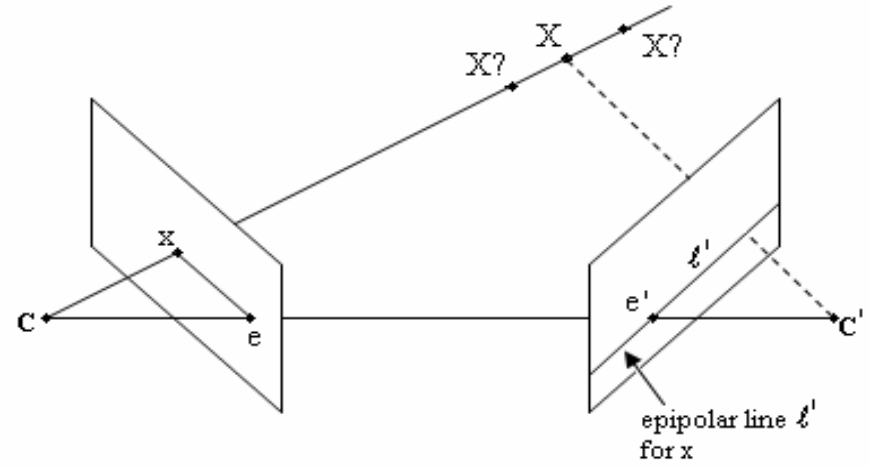


Figure 3.6 Point correspondence geometry

3.2.2 Fundamental Matrix

The fundamental matrix [24] is the algebraic representation of the epipolar geometry and it can be derived by using the mapping between the point and its epipolar line. For each point x in one image, there exists an epipolar line l' in the other image which contains the match of the point *x*, as x'. The epipolar line is the projection in the second image of the ray passing through the point *x* and camera center *C*. Hence, there is a mapping between *x* and l'; $x \mapsto l'$. This mapping can be represented by a 3x3 matrix, which is the Fundamental Matrix, F, and it is a projective mapping from points to lines.

3.2.2.1 Geometric Derivation

Geometric derivation of the fundamental matrix starts with the mapping of the point *x* to some point x' in the other image lying on the epipolar line l'. This mapping is achieved by using a plane π in space not passing through either of the two camera centers. The ray through the camera center and image point *x* intersects the plane π at the point *X*. Projection of this point X onto the other image plane is the image point x' and x' must line on the epipolar line l' as shown in the Figure 3.7. The points *x* and x' are both images of the 3D point *X* lying on a plane. Thus, there is a 2D homography H_π mapping between each x_i and x_i', as

$$x_i' = H_\pi x \tag{3.13}$$

Then, the epipolar line l' passing through the image point x' and epipolar line e' can be written as

$$l' = e' \times x' = [e']_x x' = [e']_x H_\pi x = Fx \tag{3.14}$$

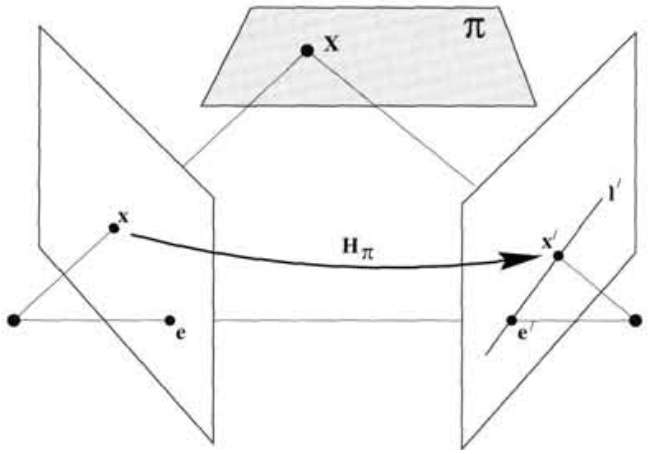


Figure 3.7 Projection of image point onto the other image plane

where $F = [e']_x H_\pi$. F is the fundamental matrix and [ ]X expression is the 3x3 skew symmetric matrix which is defined for an arbitrary vector $a = (a_1, a_2, a_3)^T$ as

$$[a]_x = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{3.15}$$

3.2.2.2 Algebraic Derivation

The Fundamental Matrix can also be derived algebraically using the two camera projection matrices P and P'. The ray back-projected from x by P is obtained by solving $PX = x$.

$$X(\lambda) = P^+ x + \lambda C \tag{3.16}$$

where λ is some scalar to parameterize the ray, P^+ is the pseudo-inverse of P satisfying PP^+=I and C its null vector, namely the camera center, defined by PC=0.

In particular, two points on the ray are P^+x (at A = 0), and the first camera centre C (at A = ∞). These two points are imaged by the second camera P' at $P'P^+x$ and $P'C$ respectively in the second view. The epipolar line is the line joining these two projected points, $l' = (P'C) \times (P'P^+x)$. The point $P'C$ is the epipole in the second image, namely the projection of the first camera centre, and might be denoted by e'. Thus,

$$l' = [e']_x \left(P'P^+\right)x = Fx \tag{3.17}$$

where $F = [e']_x P'P^+$, which is the same result with the previous one since $H_\pi = P'P^+$.

3.2.2.3 Properties of Fundamental Matrix

Fundamental matrix is a projective mapping from a point to a line. Therefore, it has a rank two and there is not any inverse mapping between the images since F is not full rank. The fundamental matrix satisfies the epipolar constraint

$$x'^T Fx = 0 \tag{3.18}$$

for any pair of corresponding points $x \leftrightarrow x'$ in the two images and maps 2D image point onto epipolar line l'.

$$Fx = l'. \tag{3.19}$$

The important properties of fundamental matrix are summarized in Table 3.1.

Table 3.1 Summary of fundamental matrix properties [24]

- F is a rank 2 homogeneous matrix with 7 degrees of freedom
- Epipolar constraint : If x and x' are corresponding image points, then $x'^T Fx = 0$
- Epipolar lines:
  - $l' = Fx$ is the epipolar line corresponding to x
  - $l = F^T x'$ is the epipolar line corresponding to x'
- Epipoles:
  - $Fe = 0$
  - $F^T e' = 0$
- Derivation of F:
  - Geometric derivation: $F = [e']_x H_\pi$ where H_π mapping each x_i to x_i'.
  - Arithmetic derivation: $F = [e']_x P'P^+$ where P^+ is the pseudo-inverse of P.

3.3 Trifocal Tensor

Trifocal tensor [24] has an analogous role in three-view geometry to that of fundamental matrix in two-view geometry. It encapsulates all the projective geometric relations between three views that are independent of the scene structure. The tensor only depends on the orientation between views and the internal parameters of the cameras and could be uniquely defined by the camera

matrices of the views. It could be computed from image correspondences alone without requiring knowledge of the orientation or calibration.

In this section, only point transfer property of the trifocal tensor from a correspondence in two views to the corresponding point in the third view will be discussed briefly. The point transfer problem is finding the corresponding point x'' in the third image, while knowing the matched pair x and x' in the first two views. This problem can be solved using the projection matrices P_1, P_2 and P_3 of three views.

As discussed in the Section 3.2.2.2, fundamental matrices F_{31} and F_{32} can be obtained by using these projection matrices. Then, corresponding epipolar lines for x and x' on the third image can be determined by using (3.19). The required point x'' must lie on both of the epipolar lines, since it is the epipolar match of point x in the first image and point x' in the second image. Taking the intersection of the epipolar lines results with the required point x'', as shown in the Figure 3.8.

$$x'' = (F_{31}x) \times (F_{32}x) \tag{3.20}$$

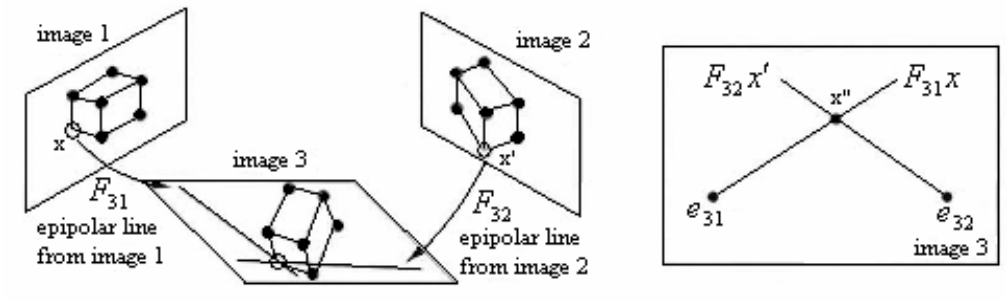


Figure 3.8 Point transfer

3.4 A Brief Introduction to Graph Theory

A graph [26] is a set of objects, called *points*, *nodes*, or *vertices* that are connected by links, denoted as *lines* or *edges*. In a proper graph, which is by default undirected, a line from point *A* to point *B* is considered to be the same thing as a line from point *B* to point *A*. Typically, a graph is depicted in diagrammatic form as a set of dots (for the points, vertices, or nodes), joined by curves (for the lines or edges). [26], as shown in Figure 3.9.

Given a connected undirected graph, a spanning tree [27] of that graph is a subgraph which is a tree and connects all its vertices together. A single graph might have many different spanning trees. A weight to each edge can also be assigned, which is a number, representing how unfavorable it is, and can be used to assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree. A minimum spanning tree (or minimum weight spanning tree) is then a spanning tree with weight less than or equal to the weight of every other spanning tree. More generally, any undirected graph has a minimum spanning forest [27].

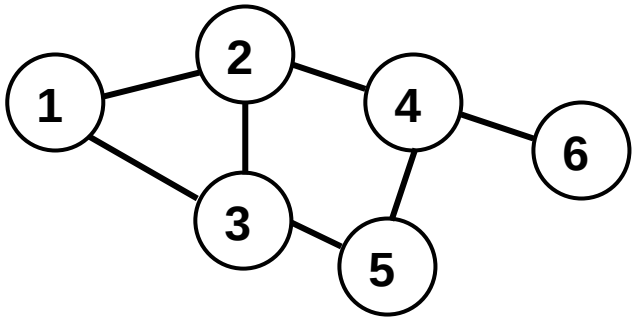


Figure 3.9 A labeled graph on 6 vertices and 7 edges.

The minimal spanning tree can be found by using Kruskal's algorithm [26]. Kruskal's algorithm is an algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. In other words, it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component). Kruskal's algorithm is a typical example for a greedy algorithm and can be summarized as follows [26]:

1. Create a forest *F* (a set of trees), where each vertex in the graph is a separate tree
2. Create a set *S* containing all the edges in the graph
3. While *S* is nonempty
4. Remove an edge with minimum weight from *S*
5. If that edge connects two different trees, then add it to the forest, combining two trees into a single tree
6. Otherwise discard that edge

At the termination of the algorithm, the forest has only one component and forms a minimum spanning tree of the graph.

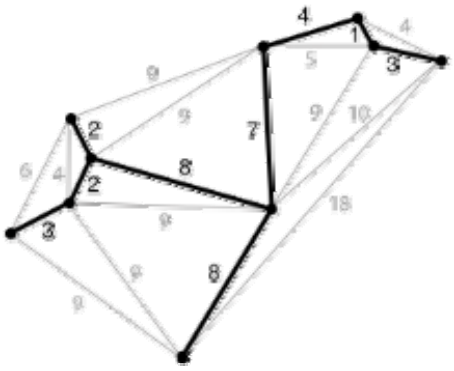


Figure 3.10 The minimum spanning tree of a planar graph. Each edge is labeled with its weight, which here is equal to its length.

CHAPTER 4

MULTI-CAMERA BACKGROUND MODELING

Foreground object detection via background subtraction has been used extensively in video surveillance applications due to its satisfactory performance and computational effectiveness. However, when a single camera is used, it is relatively difficult to deal with false segmentation results caused by dynamic scenes and shadows. These false segmentations result in failure in subsequent actions, such as tracking and event recognition.

The use of multiple cameras leads to better handling of dynamic scenes, shadows and illumination changes due to the utilization of 3D information, compared to a single camera. However multi-camera methods usually have heavy computational load with respect to the previously discussed single camera methods.

Most of the methods employing multi-camera systems, which are discussed in the following section, use stereo cameras and depth information to model their background. However, stereo cameras are not available in most of the surveillance applications, and systems consisting of individual cameras with intersecting field of views (FOVs), as shown in Figure 4.1, are generally preferred, due to their wide area coverage.

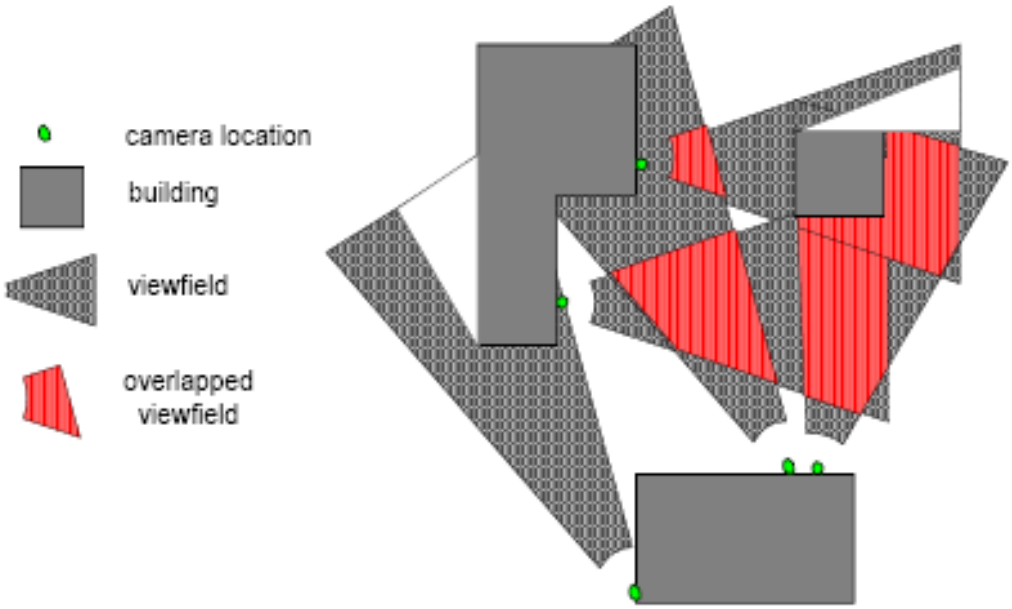


Figure 4.1 An example of camera lay-out in surveillance applications [28]

In this chapter, multi-camera background modeling methods, which are comparatively less demanding in terms of computation, are discussed. These methods use the information coming from two separate cameras with intersecting FOVs, that is obtained by relating the input images by the homography matrix. The incoming information from the cameras is concatenated by using various simple fusion methods and therefore, they might achieve real-time performance. The simulation results are presented at the end of each section.

4.1 Related Work

Many methods have attempted to solve some of the background modeling problems by using multi-camera configurations. Most of them use stereo cameras to segment the foreground regions. Goldlucke and Magnor [30] present an

algorithm to reconstruct the 3D-geometry of a static scene by using images captured from a number of calibrated cameras directly. A depth value is assigned to each pixel which defines its location in 3D-space; hence, a 3D background model is presented. However, in case of a stereo camera, it is necessary to find the stereo correspondences precisely, to model the background. Thus, stereo algorithms can be used, as proposed in [31][32]. Goldlucke and Magnor assume that they have a set of fully calibrated cameras and an image of the static background for each camera with at least approximate per-pixel depth information. However, these assumptions are difficult to satisfy, when wide-baseline camera configurations are considered.

Ivanov et al. [33] described a method that employs the accurate stereo, for constructing the background model, where the color difference between conjugate pixels is used to distinguish between background and foreground. Hence, the on-line computation of depth and the reconstruction of the 3D model of space at each step are avoided. Their method is based on the fact that if the background is geometrically static, stereo disparity between a primary and an auxiliary camera views of the background scene is also static, fully specifying the pixel-to-pixel transformation from one image of empty scene to another. This method has the obvious drawback of pixel-to-pixel stereo disparity calculation in wide-baseline systems with dynamic background.

Lim et al. [34] propose a sensor configuration that eliminates false detections of the method proposed in [33]. They claim that the performance of the method discussed in [33] is improved by eliminating most detection errors due to missed detections, specular reflections and objects being geometrically close to the background. However, their method still has the drawbacks of [33] when wide-baseline configurations are considered.

Harville et al. [29] proposed a multimodal system, which adapts Gaussian mixture models of the expected background appearance, to the combined image

feature space of depth and luminance-normalized color. They use spatially-registered, time-synchronized pairs of color and depth images that are obtained by static cameras. This method has quite interesting results. However, the complexity of the algorithm is relatively high and the depth information for the observation vector of each pixel is, again, difficult to find for camera configurations for wide-baseline systems.

4.2 Background Modeling from Multi-view Video

Multi-camera background modeling algorithms give promising results, considering the previous research efforts. Many false segmentation results which occurred due to illumination variances and dynamic background objects, are eliminated by using the fused information coming from the multi-sensor network. However, depth information is still not available for most of the surveillance applications.

The simplest way of segmenting foreground regions by using multi-view system is fusing the incoming images via homography between images. Homography relation is explained in Section 4.3 and two different foreground decision methods are also discussed in Section 4.4 and 4.5.

Harville et al. [29] is pursued for developing a background modeling algorithm due to their promising results with relatively simple algorithm. In their method, a common background model for two-views is constructed by using mixture of Gaussians. However, instead of constructing the observation vector by color and depth information, which is relatively hard to obtain, RGB values of matched (by homography) pixels can be used. This proposed method is described in detail at Section 4.5.

4.3 Relating Images With Homography

Information coming from the different camera systems must be related to each other to generate a common framework. Camera calibration and fundamental matrix can be used to relate them; however, automated camera calibration is a difficult task, when typical wide-baseline camera systems are considered. It is difficult to find the epipolar matches and the 3D coordinates of the features/objects, and 3D computations demand heavy computational load.

In this chapter, instead of fundamental matrix, homography matrix is used to relate the images to each other which are taken by different camera configurations. Homography matrix H_{12} between two camera views, assuming that the region of interest is planar (ground-plane assumption), can be derived by using the matched points x_i and x_i' as shown in the Figure 4.2. Homography matrix calculation is explained in detail in the Section 3.1.5.

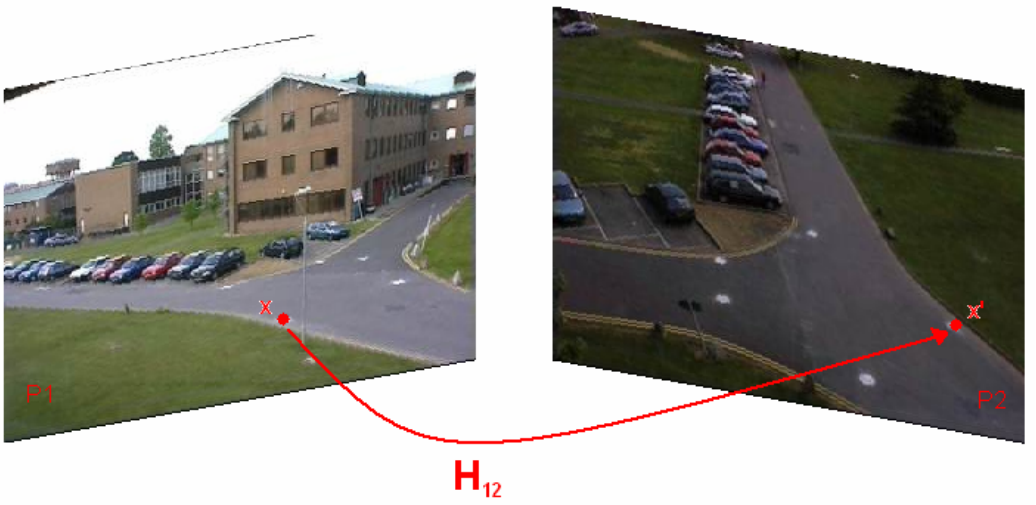


Figure 4.2 Corresponded points used to find homography matrix

Moreover, the common region of interest (ROI) for the camera pair must be defined to specify the region in which the cameras will cooperate. In the regions excluding common ROI, cameras perform their single camera tasks. This region of interest is defined by the intersection of the FOVs of cameras and it is obtained by using the homography matrix.

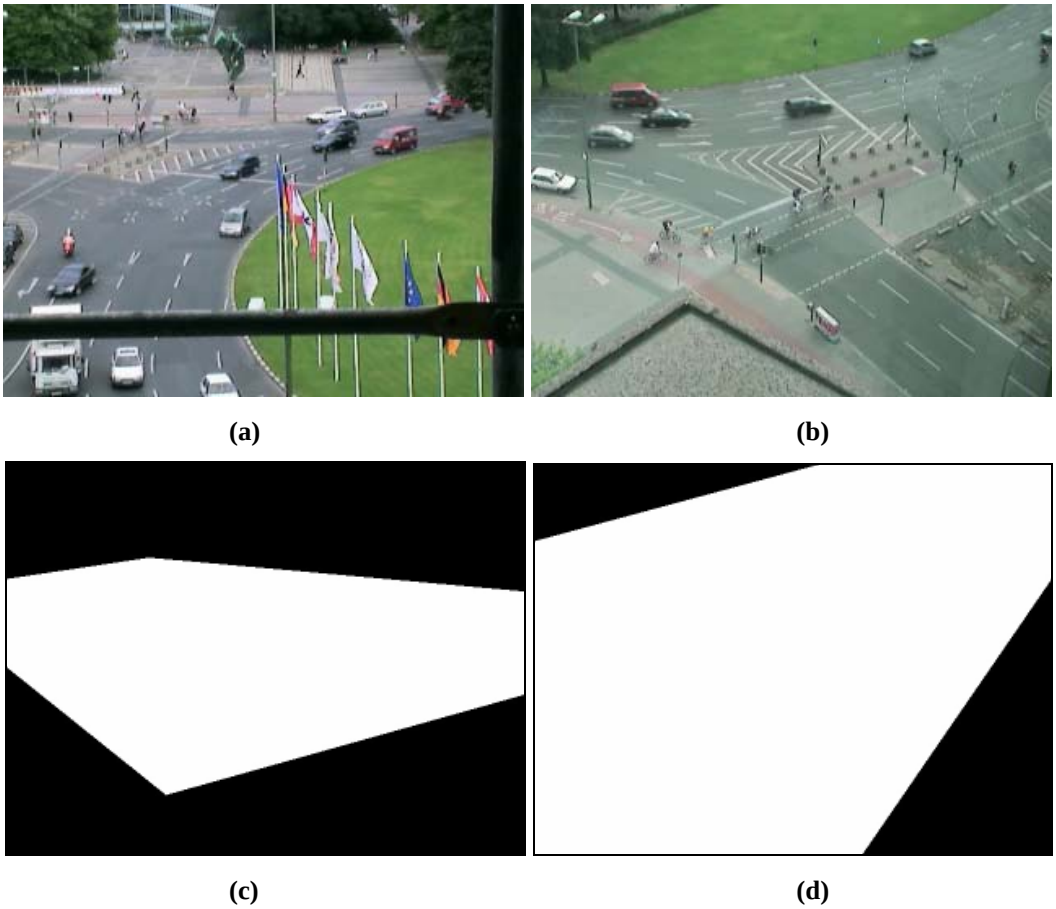
Projections of all pixels in the first camera view onto the second camera view, which are calculated by using the homography matrix, can be used to find the intersecting FOV. Projected pixels which are inside the boundaries of second camera's image plane are the pixels of intersecting FOV. It is illustrated in the Figure 4.3 and Figure 4.4.

In the following sections, the homography matrix is used by planar object assumption in which the foreground objects are assumed planar and lie on the ground plane. Furthermore, in such two-camera systems, one of the cameras is defined as reference camera, whereas the other one is defined as auxiliary camera.

4.4 Foreground Detection by Unanimity

One of the primary decision methods, which can be used to integrate incoming information, is checking the unanimity of foreground-background decisions of the two cameras. Final decision is given according to the individual decisions of both cameras, if a certain foreground pixel in the first camera view is also labeled as a foreground pixel in the second camera view, then it is decided as foreground in the final decision.

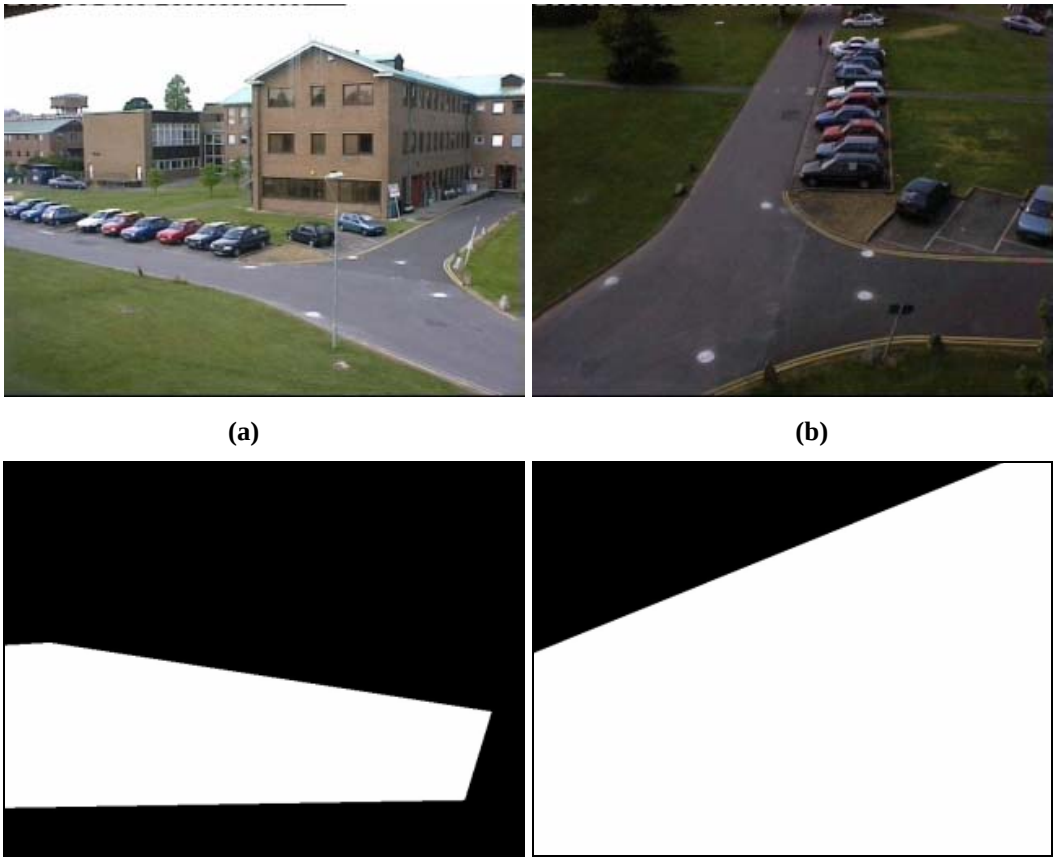
In this method, each camera has its own background model, and performs its background subtraction independently by using one of the background modeling algorithms discussed in Chapter 2. Next, homography matrix H_{12} between the



(a) **(b)**

(c) **(d)**

Figure 4.3 Related images of "junction data" (a) and (b) are the input images of first and second cameras respectively, (c) and (d) are the defined common FOVs



(a) **(b)**

(c) **(d)**

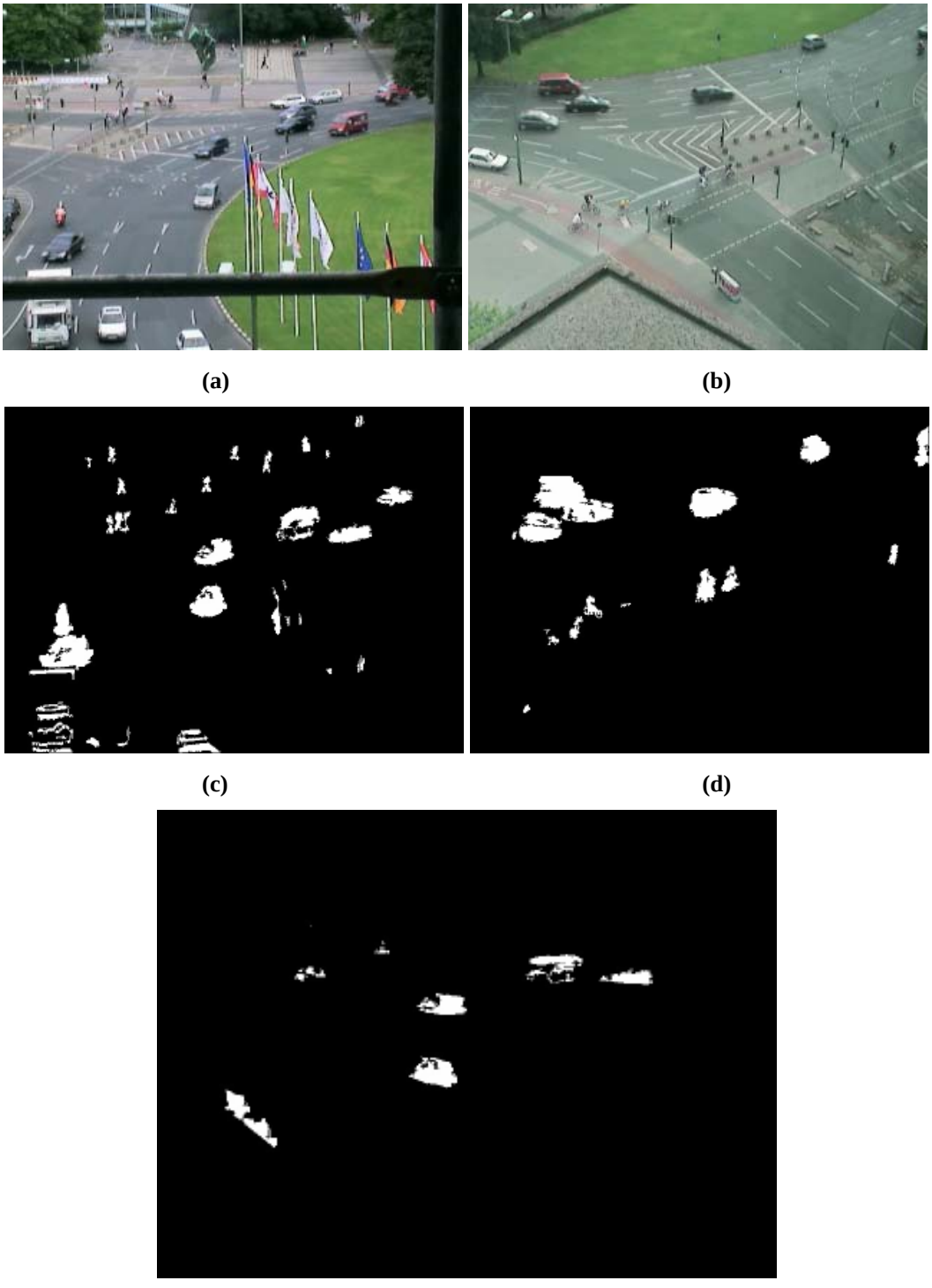
Figure 4.4 Related images of "PETS2001 dataset1" [76] (a) and (b) are the input images of first and second cameras respectively, (c) and (d) are the defined common FOVs

first and second camera is calculated. After foreground masks are generated for each camera, foreground pixels in the first camera's intersecting FOV are checked by the foreground pixels in the second camera's intersecting FOV region.

For each foreground pixel x_i in the first camera's ground plane (intersecting FOV), corresponding point x_i' in the second camera's ground plane is determined. If the point x_i' is also labeled as foreground pixel by the second camera, then x_i remained untouched, otherwise, it is labeled as background. The foreground pixels which are out of the common FOV, are processed by using single camera algorithms.

This method eliminates the false detected regions in the first camera view which are segmented correctly in the second camera view. As shown in Figure 4.5, false segmented regions due to the flags are eliminated. In this simulation, foreground pixels, which are outside the common FOV, are masked to show only the results of discussed algorithm.

The most important drawbacks of the proposed system are the projection errors that occur due to the plane object assumption and the segmentation errors in the second camera view. Any error in the second camera view directly affects the final decision and masks the foreground regions of the first camera. Therefore, a method based on the weighting of the decisions is proposed in the next section.

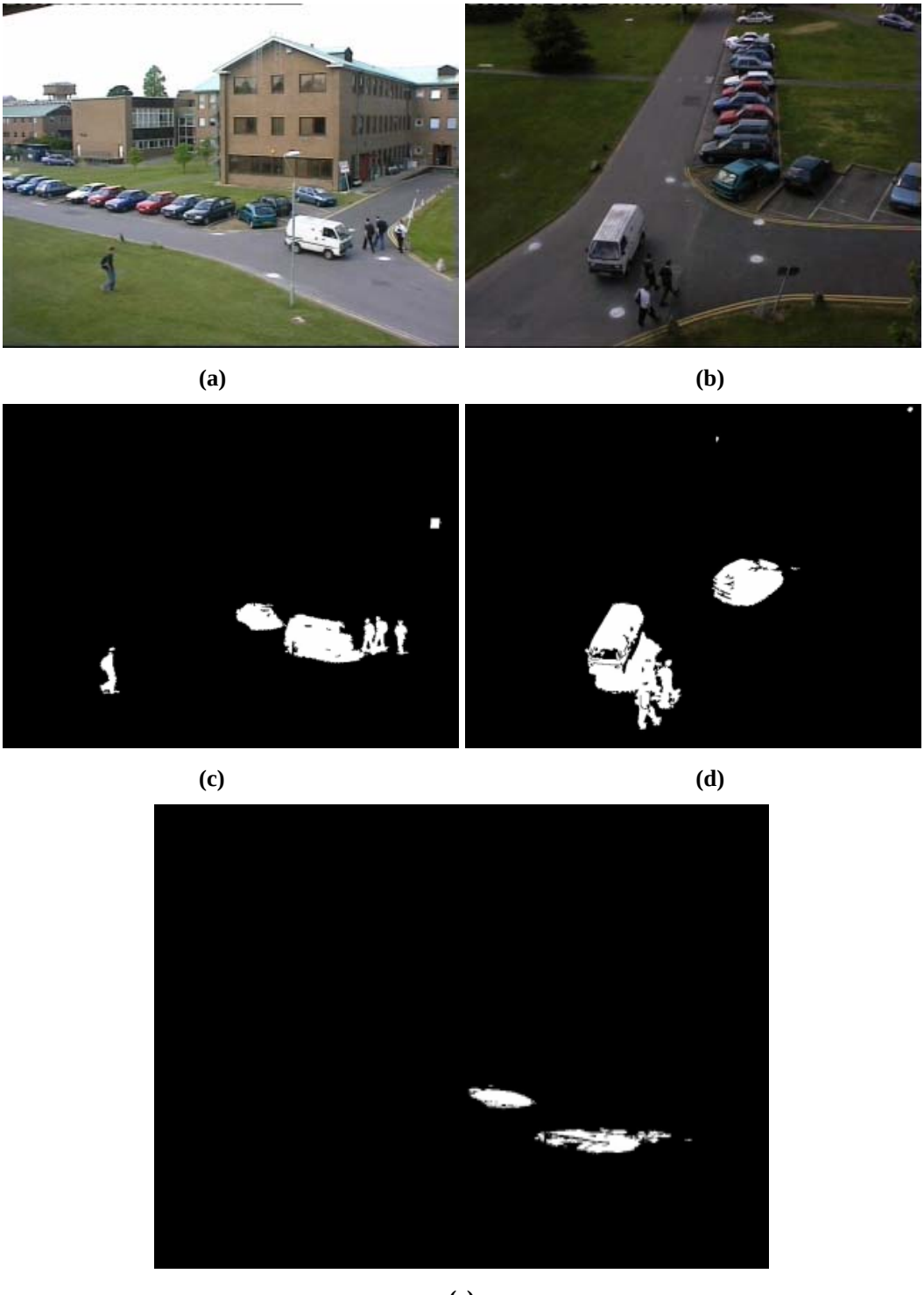


(a) **(b)**

(c) **(d)**

(e)

Figure 4.5 Simulation results of "junction data" (a) and (b) are input images, (c) and (d) are corresponding foreground masks obtained by using MOG method, (e) is the final foreground mask



(a) **(b)**

(c) **(d)**

(e)

Figure 4.6 Simulation results of "PETS2001 dataset1" [76] (a) and (b) are input images, (c) and (d) are corresponding foreground masks obtained by using MOG method, (e) is the final foreground mask

4.5 Foreground Detection by Weighted Voting

In multi-camera configurations, different parts of the scene may be monitored more clearly by different cameras due to their positions. Therefore, utilization of the decisions of cameras by equal weights is not suitable in some cases. In the proposed method, instead of checking unanimity, a similar method for "voting with weights" is used. Each camera votes for a common FOV pixel with a certain weight whether it is a foreground pixel or not. The weight of the vote for the pixel is decided with respect to the corresponding pixel's background model. If the pixel differs strongly from its background model, it is voted with larger weight as foreground. Moreover, one of the cameras with a better positioning is decided as the reference camera and it has a larger weight with respect to the other (auxiliary) camera. The final decision is given by adding the votes and comparing the sum with a certain threshold.

Similar to the previous method, each camera has its own background model. For each pixel in the first camera view x_i, corresponding point x_i' in the second view is obtained by using a homography. Then, for each pixel in the pair $x_i \leftrightarrow x_i'$, its difference with the corresponding background model is obtained. Next, their votes are weighted by using (in proportion) the calculated differences and corresponding camera's weight. After this step, the weighted votes are added and resulting sum is compared against a certain threshold and the foreground mask is determined, as formulated in (4.1).

$$\begin{aligned} BD(x_i, x_i', t) &= \alpha \left| I(x_i, t) - BM_i(x_i, t-1) \right| + \beta \left| I(x_i', t) - BM_i'(x_i', t-1) \right| \\ FM(x, y, t) &= \begin{cases} 1 & BD(x_i, x_i', t) > threshold \\ 0 & BD(x_i, x_i', t) < threshold \end{cases} \end{aligned} \tag{4.1}$$

where $I(x_i, t)$ and $I(x_i', t)$ are the intensity values of the pixels in the matched pair at time t, $BM_i(x_i, t-1)$ and $BM_i(x_i, t-1)$ are the corresponding pixels'

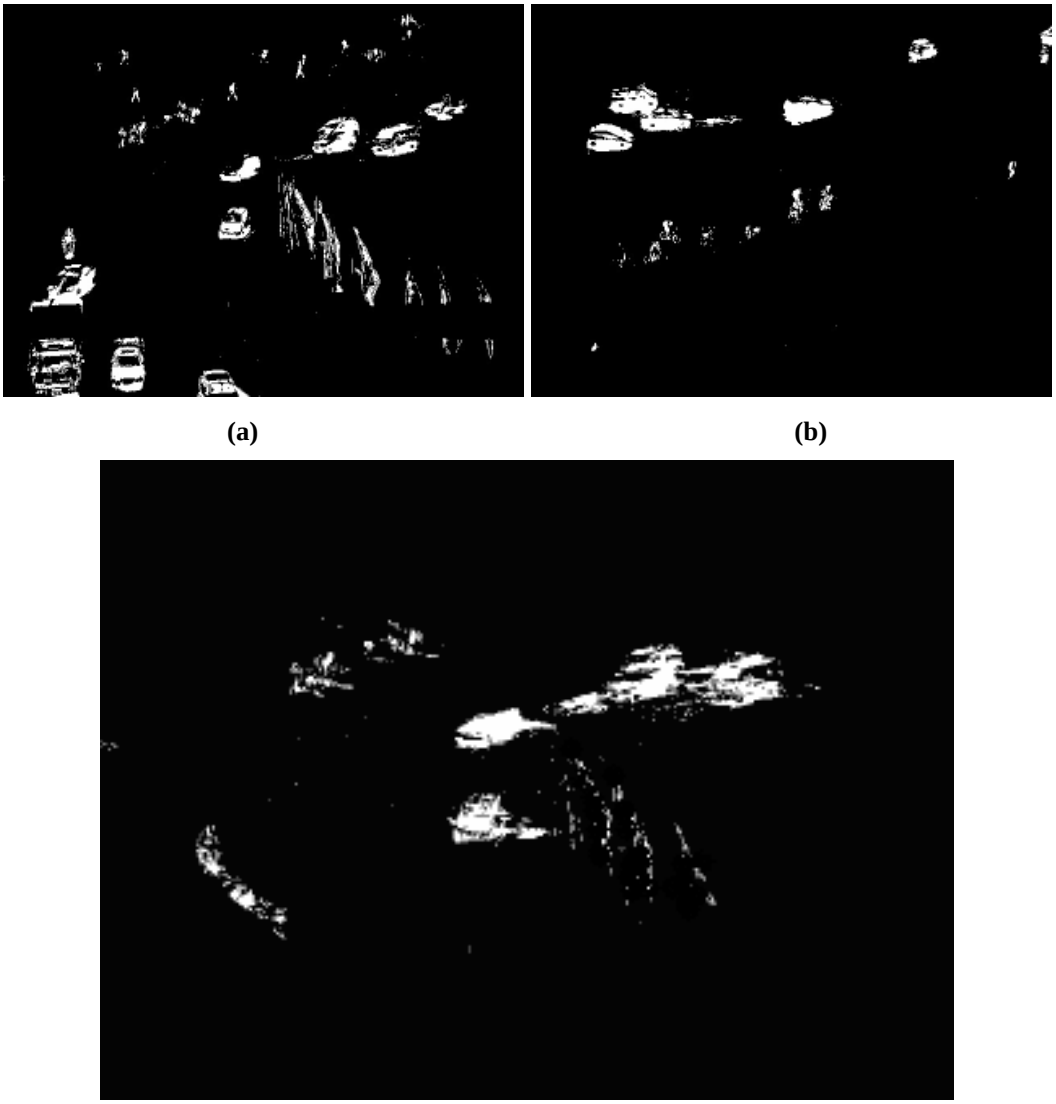
background models at time t-1, α .and β are the coefficients to adjust the contributions of the cameras. Generally, the contribution for the first camera (reference camera with better positioning) is larger than the second one, and $\alpha \geq \beta$.

As a result, weak vote (with small weight), given by one of the cameras as foreground for a certain pixel, is suppressed, if the decision, given by the other camera for the corresponding pixel as background, is strong (large weight) or vice versa. Moreover, two weak foreground votes might result in foreground decision, if their sum is large enough.

Frame differencing is performed to find the foreground masks, as shown in Figure 4.7 (a) and (b), and segmented foreground pixels, which are outside the common FOV, are masked to show only the results of discussed algorithm. As shown in the Figure 4.7 (c), false segmented regions due to the flags are mostly eliminated, while the segmented vehicle masks are preserved. However, plane-object assumption also adds some projection error noise to the segmented foreground masks.

4.6 Mixture of Multivariate Gaussians Background Model

The proposed methods that are considered up to this point, are based on segmentation of the foreground regions by using two camera models. Hence. each input image pair is first compared with its corresponding background models. Then, decisions or comparison results of each camera are fused to give the final decision. However, these methods are very sensitive to this final decision method and votes given by cameras can be unified by several different methods. For example, the final decision can be given by taking the majority of votes into account, or by just unanimity. However, each decision method is suitable for a different case and not appropriate for an overall system.

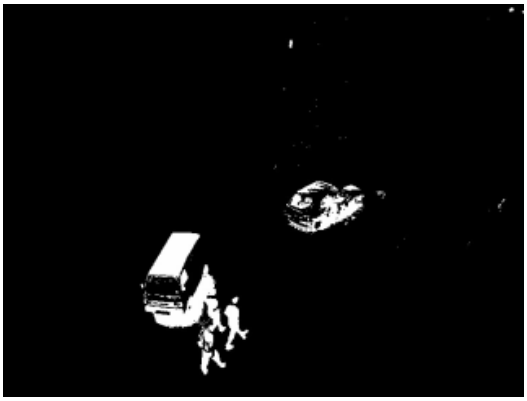


(a) **(b)**

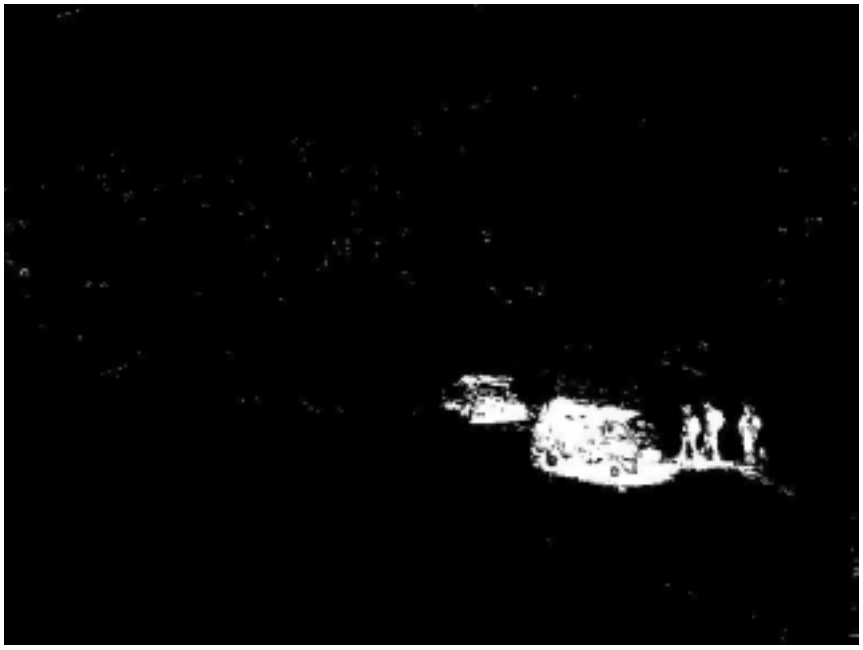
(c)

Figure 4.7 Simulation results of "junction data" (a) and (b) are corresponding foreground masks obtained by using frame differencing method, (c) is the final foreground mask

(a)



(b)



(c)

Figure 4.8 Simulation results of "PETS2001 dataset1" [76] (a) and (b) are corresponding foreground masks obtained by using frame differencing method, (c) is the final foreground mask

Hence. a multi-camera background modeling technique, which is similar to the one discussed in [29], by using mixture of multivariate Gaussians, is proposed.

Multivariate Gaussians can be thought of as a generalization to higher dimensions of the one-dimensional Gaussians. Therefore, mixture of multivariate Gaussians background modeling can be thought of as a generalization to higher dimensions of the single camera mixture of Gaussians background modeling [3]. Each dimension is constructed by using the information coming from one of the cameras. For a two camera case, which is discussed in this thesis, the first camera and the second camera construct a single background model

Each pixel in the main camera view and intersecting FOV's unified background model is constructed by mixture of K multivariate Gaussian distributions. The pixels in the non-overlapping field of the main camera view are modeled by using mixture of one dimensional Gaussians, which is similar to the single camera case. Then, the probability of observing a pixel value X_N in the common FOV at time N is

$$p(X_N) = \sum_{k=1}^{K} w_k \frac{1}{(2\pi)^{D/2}\left|\sum_k\right|^{1/2}} e^{-\frac{1}{2}(X_N-\mu_k)^T \sum_k^{-1}(X_N-\mu_k)} \tag{4.2}$$

where $X_N = \begin{bmatrix} x_N \\ x'_N \end{bmatrix}$ and $\mu_k = \begin{bmatrix} \mu_{k_1} \\ \mu_{k_2} \end{bmatrix}$ and $\sum_k = \begin{bmatrix} \sigma_1^{\,2} & \sigma_1\sigma_2 \\ \sigma_1\sigma_2 & \sigma_2^{\,2} \end{bmatrix}$ and w_k is the weight of k^{th} Gaussian.

In the formulation above, x_N refers to a pixel in the main camera's intersecting FOV, while x'_N is the homographic projection of x_N onto the auxiliary camera's intersecting FOV.

Then

$$x'_N = H_{12} x_N . \tag{4.3}$$

Finally, for each pixel x_N, there are 3 mixtures of K multivariate Gaussian models, each corresponding to a color channel, R, G or B. For other pixels, formulation that is explained in the Section 2.1.1.4 is used.

Every new pixel value, x_t, is checked against the existing K Gaussian distributions, until a match is found. A match is defined as a pixel value within 2.5 standard deviations of a distribution. If none of the K distributions match the current pixel value, the least probable distribution is replaced with the current value as its mean value, an initially high variance, and low prior weight.

The prior weight of the k[th] Gaussian at time t is adjusted as follows

$$w_{k,t} = (1-\alpha) w_{k,t-1} + \alpha M_{k,t} \tag{4.4}$$

where α is the learning rate and $M_{k,t}$ is '1' for matched models, and '0' for remaining models.

The μ and σ for unmatched distributions remain same, while the matched ones are updated as follows;

$$\begin{aligned} \mu_t &= (1-\varphi)\mu_{t-1} + \varphi X_t \\ \sum_t &= (1-\varphi)\sum_{t-1} + \varphi (X_t - \mu_t)(X_t - \mu_t)^T \end{aligned} \tag{4.5}$$

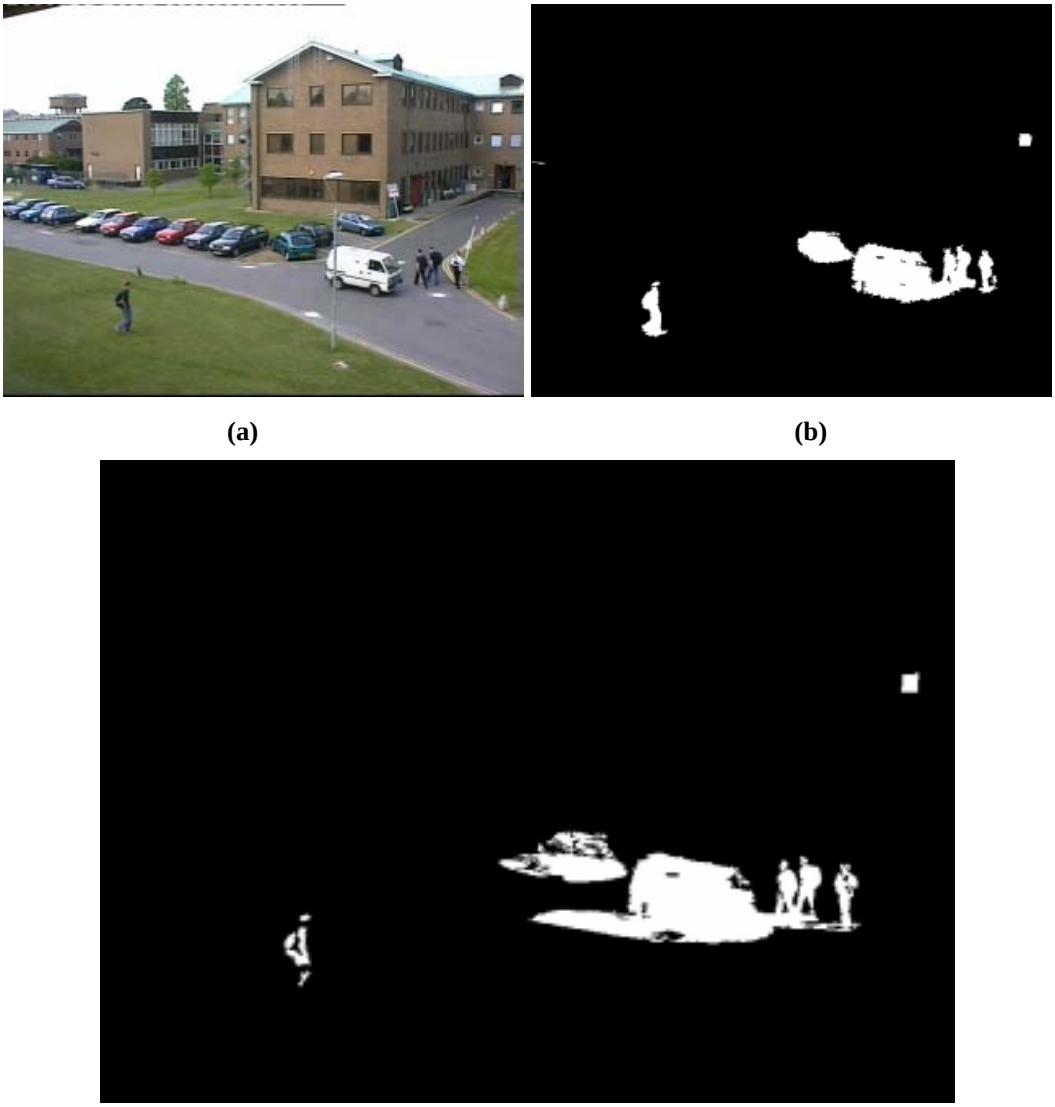
where $\varphi = \alpha \eta(X_t \mid \mu_k, \sigma_k)$.

(a)

(b)

(c)

Figure 4.9 Simulation results of "junction data", (a) input frame (b) corresponding foreground mask found by using single camera MOG, (c) foreground mask found bu using mulltivariate MOG



(a) **(b)**

(c)

Figure 4.10 Simulation results of "PETS2001 dataset1" [76], (a) input frame (b) corresponding foreground mask found by using single camera MOG, (c) foreground mask found bu using mulltivariate MOG

This method suffers from projection errors due to the assumption that objects are planar and, in some cases, in which the tall vehicles are considered; these errors result in larger foreground masks than the actual size of the objects. However, some missed vehicles in single camera MOG method can be segmented in multivariate MOG method, as shown in the Figure 4.9, and the proposed method also has real time performance.

4.7 Conclusion

In all of the discussed methods, the projection errors have important effect on the false segmented foreground masks. The assumption that the moving objects are considered planar, results erroneous masks especially for large vehicles. Therefore, these methods and homography matrix usage to relate images are more suitable for systems in which the cameras are mounted on devices at high altitudes, where the height of the object becomes negligible with respect to the altitude of the camera. If the camera lay-out is as suggested, the projection errors are minimized and the proposed method could give more precise foreground masks.

CHAPTER 5

MULTI-CAMERA OCCLUSION HANDLING

Occlusion is one of the primary problems of single camera surveillance systems. In the moving object detection process, occlusions cause moving objects to be either segmented in an erroneous shape or became totally lost. These false segmentation results cause subsequent actions to fail or a decrease in their performance.

Occlusions could be separated into two main classes. The first one is *static occlusion* where the foreground object is occluded by an obstacle in the background scene. In partial static occlusion, moving object can still be tracked with a loss in size and shape information. However, in the total occlusion cases, track of the occluded object is generally lost and must be predicted within the frames in which the occlusion has occurred. In the second type, which is called *dynamic occlusion*, a moving object occludes another moving object. In partial dynamic occlusions, occluded objects can be separated by using some prior information about the objects, such as their size or shape, or the background environment, like the road lines. However, in total dynamic occlusions, separation methods do not succeed and tracking is performed by prediction.

The use of multiple cameras leads to better handling of occlusions compared to a single camera, due to the utilization of 3D information and the presence of

different points of views. This information can be gathered to form an occlusion-free result.

In this chapter, an occlusion handling method by using multi-camera (2 camera) system with intersecting FOVs is explained. First, various related work in the literature is briefly discussed. Then, performed segmentation algorithms and the point transfer method are explained. Finally, point segmentation method is discussed.

5.1 Related Work

There are several methods for single camera surveillance with some occlusion handling, but all have limited performance, especially in severe occlusions. Haritaoglu et al. [1] propose a method to count and track people in groups with occlusions. System use local shape information by analyzing the boundary of the object, and try to find pieces that are similar to human heads based on local shape approximations to the object boundary. Moreover, global shape information and constraints, derived from the requirement that the head to be aligned with the axis of the torso are used to locate torso as well. Then, each detected head is tracked by using correlation-based matching. However, their method necessitates correct initial masks to locate people and create their appearance models, which is difficult to satisfy in severe occlusions and noisy foreground masks.

Tekalp and Dockstade [35] use combination of the tracking results from previous frames with motion estimates generated from the current frame to split regions by a clustering routine. Allowing regions to split provides the Kalman filtering process with a set of observations for occluding and occluded objects. Nevertheless their method has the obvious drawback of handling (tracking) initially occluded objects.

Pang et al. [36] employed the texture-based vehicle-segmentation approach to extract the moving vehicles for segmentation. Based on the segmented shape of the object and due to the reason that the shape can be represented by a simple cubical model, they propose a two-step method: first, detection of the curvature of the shape contour to generate a data set of the occluded vehicles and, second, decomposing it into individual vehicle models by using a vanishing point in three dimensions and the set of curvature points of the composite model. This method is strongly dependent to the extracted foreground mask and therefore very sensitive to the segmentation noise.

Xuefeng and Nevatia [37] use generic shape models of classes of vehicles, such as for a sedan or an SUV, to detect, classify and track vehicles from motion foreground blobs. They assume that these blobs are generated by moving vehicles or by noise. Shadows and reflections are assumed to be absent or removed by some pre-processing. However, these assumptions are difficult to satisfy and the complexity of the algorithm is relatively high.

Apart from the aforementioned methods, feature-based trackers are also used to handle occlusions where partial occlusions relatively solved [18][38]. However, segmentation of the features into individual objects, generally fail in dynamic occlusions.

On the other hand, multi-camera based surveillance systems have improved occlusion handling capabilities with respect to single camera systems. Krumm et al. [39] present an algorithm that information from stereo cameras is combined in 3D space. After performing background subtraction, human-shaped blobs in 3D space are detected. In their method, color and other information is used to identify and track people over time. However, stereo cameras are not available in most of the surveillance systems and 3D calculations are computationally demanding.

M2Tracker [41], which is similar to the method in [40], used a region-based stereo algorithm to find 3D points inside an object, and Bayesian pixel classification with occlusion analysis to segment people occluded in different levels of crowd density. The complexity of this algorithm is relatively high due to the pixel-wise classification.

Orwell et al. [43] propose a color tracking algorithm to track multiple objects via multiple cameras. Connected blobs obtained from background subtraction are modeled by using color histogram techniques and are used to match and track objects. However, their method is also dependent to the initial masks of the object and lacks a solution for initially occluded objects.

Kim and Davis [40] propose a method, a multi-view multi-hypothesis approach, for segmenting and tracking multiple persons (possibly occluded) on a ground plane. In order to more precisely locate the ground location of a person, all center vertical axes of the person across views are mapped to the top-view plane and their intersection point on the ground is estimated.

Also in [42], Mittal and Davis describe an algorithm for detecting and tracking multiple people in a cluttered scene using multiple synchronized cameras located far away from each other. They over-segment each image, and use these segments to compare regions across the views along epipolar lines. The centers of the matching segments are projected onto the ground plane to identify 3D points in the scene which are potentially corresponding to people. The results from cameras are then combined to estimate the 2D locations of the people. These estimates are then used to track people across time. These two interesting methods [40][42] have quite acceptable experimental results.

5.2 Occlusion Handling from Multi-view Video

Considering the previous research attempts in the literature, occlusion handling in multi-camera settings is quite promising, in which the information from two (or more) sources are fused appropriately to eliminate the effects of occlusions in at least one of their views. In this thesis, the approach by Mittal and Davis [42] is pursued as the occlusion handling algorithm due to their promising results with their relatively simple algorithm.

The simplest way of handling the occlusion problem is to generate (virtual) views of the scene that are free of occlusions from multi-view data. However, the rendering process is a computationally demanding procedure for each pixel; hence, such methods should be applied in a simpler manner by utilizing small regions, instead of pixels [42]. Typical segmentation algorithms are examined and compared in Section 5.3 for this purpose.

Following the region-based modeling, the regions should be transferred to occlusion-free views so that the tracking of the objects could be obtained better. In Section 5.4, point transfer via trifocal transfer is presented in detail.

Since the objects are converted into sparse set of regions (rather than a dense pixel set), the segmentation problem of those points into different objects in the top view should still be solved. A clustering-based approach is presented in Section 5.5 as a solution alternative.

Proposed algorithms are tested using various data sets. Typical two results and conclusion is given in Section 5.6.

The overall method is shown in Figure 5.1 as a block diagram. The details of this method are given in the following sections.

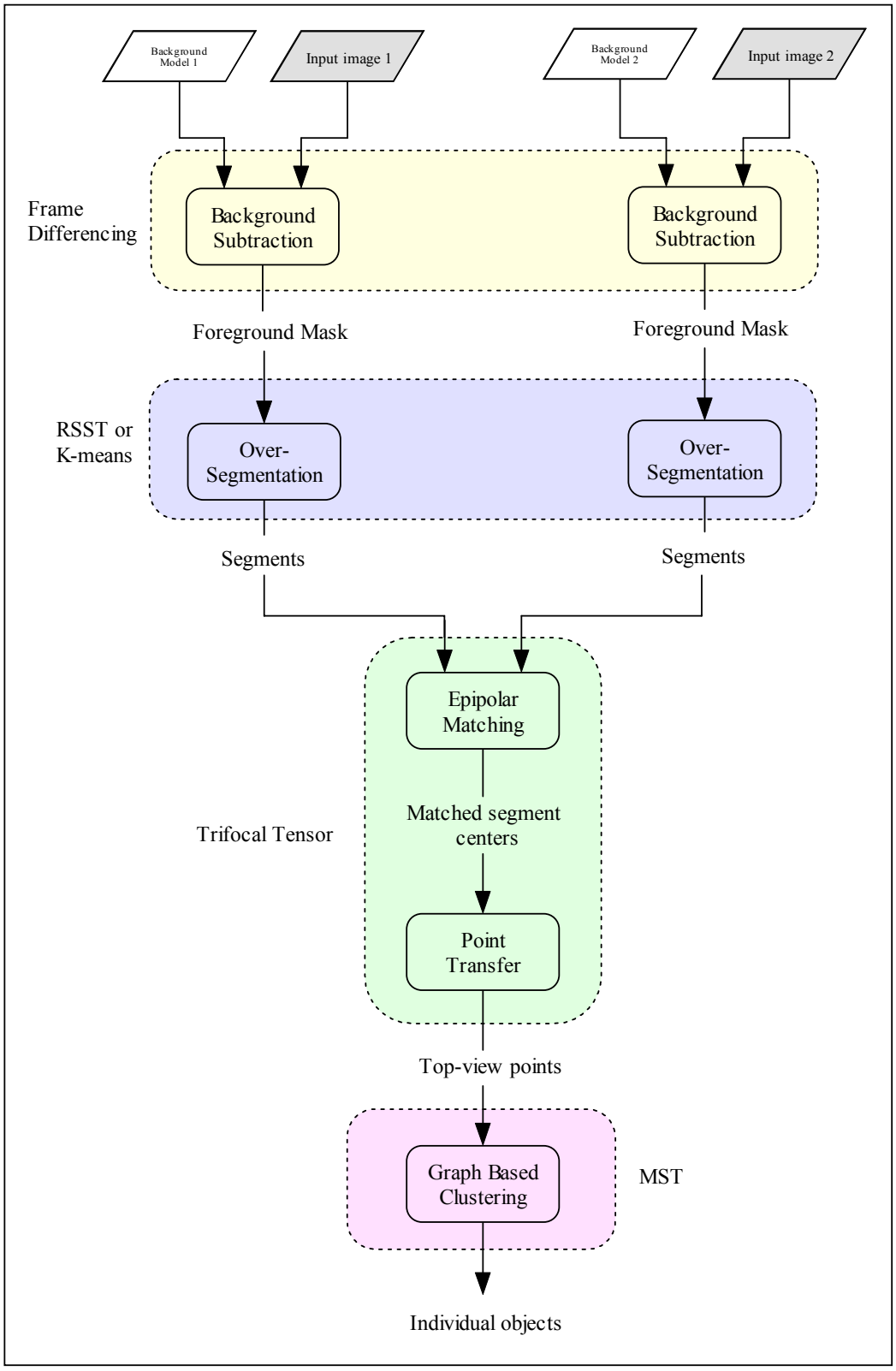


Figure 5.1 Block diagram of the proposed method

5.3 Graph-Based Segmentation

The occlusion-free view generation should only be performed for foreground regions, which consist of moving objects. Background model for each of the two views could be constructed separately by using any of the background modeling algorithms that are presented in Chapter 2. Then, background subtraction is performed for the incoming frames to extract foreground regions.

After moving object detection stage, foreground regions are segmented for partitioning these objects into homogeneous regions by the help of the features of pixels extracted from the image. These features are the intensity values of the pixels and their coordinates. As a result, the pixels that are close to each other with similar intensities are grouped together. However, in order to obtain more precision, the segmentation step is performed so that every foreground object will consist of many small regions. These small segments are used to replace the pixel-based view rendering process, while finding the epipolar matches in two views.

Two popular algorithms are tested for the segmentation of the foreground regions, which are Recursive Shortest Spanning Tree (RSST) [44] and K-Means Algorithm [22].

5.3.1 Recursive Shortest Spanning Tree

RSST [44] algorithm starts by a mapping from the image onto a weighted map. During the start up, each pixel corresponds to a different vertex or region in the weighted graph. Vertex values and link weights of the weighted graph are calculated by using the intensity values of the image pixels. A vertex value is defined as the average intensity value of the corresponding region, whereas a link

weight is evaluated by a link weight function or a cost function, which is basically a function of the vertex values and the sizes of the connected regions.

Then, all links are sorted based on their link weights and a link with the least link weight in the graph is selected to be the next link of the shortest spanning tree (SST). The chosen link is saved and the connecting regions are merged. The vertex values of the newly merged region is updated, hence, all surrounding links need to be recalculated and all loop-forming links, also known as duplicated links, will be removed. Subsequently, all remaining links are sorted. Thus, the number of regions is progressively reduced from M x N in an M pixel by N pixel image, down to just one region, if desired. Those saved links form a spanning tree representation of the image [44]. By noting the order in which the links are saved, hierarchical representation of the original image can also be created

5.3.2 K-Means Clustering

The main idea of K-means clustering [22] is to find the k mean vectors $\mu_1, \mu_2, ..., \mu_k$ that represent all the intensities of the image in the most efficient way, where k is the number of cluster centers. By choosing those mean vectors, which include the intensity and coordinate information of the image pixels, it is also possible to segment images. It is typical to randomly choose initial cluster centers [22]. The algorithm is then summarized as

1. Initialize n, k, $\mu_1, \mu_2, ..., \mu_k$
2. Classify n samples according to the nearest μ_i
3. Recompute μ_i
4. Return step 2 until no change in μ_i

5.3.3 Segmentation Results

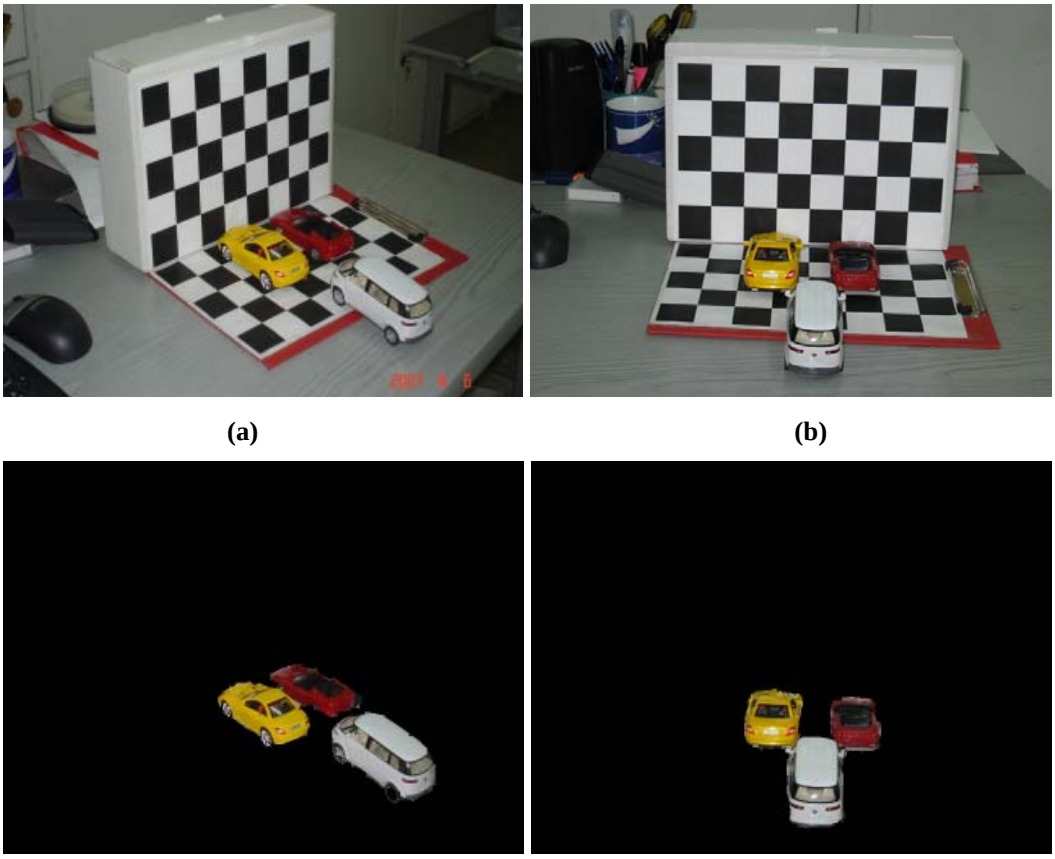
Foreground regions are detected by using simple frame differencing algorithm and filtered by using morphological operators, as shown in the Figure 5.2.

Detected foreground regions are over-segmented by two methods to test their performance and the overall performance, as shown in the Figure 5.3 and Figure 5.4.

The results for both segmentation methods are found out to be applicable for epipolar matching. The number of segments has an important role for the occlusion handling method. If the number of segments is relatively small, such as 300-500 segments, generated view does not represent the foreground objects. When the number of segments is equal to, for example between 700-1000 segments, the proposed algorithm yields better results. However, the time to segment foreground regions increases by increasing the number of segments. Moreover, the area of the foreground regions affects the computational load of the algorithms. These two algorithms are implemented in C++, and tested on a computer with Pentium-IV processor and 1 GB RAM with two sets of images with different sizes that are tabulated in Table 5.1.

Table 5.1 Time measurements of segmentation algorithms

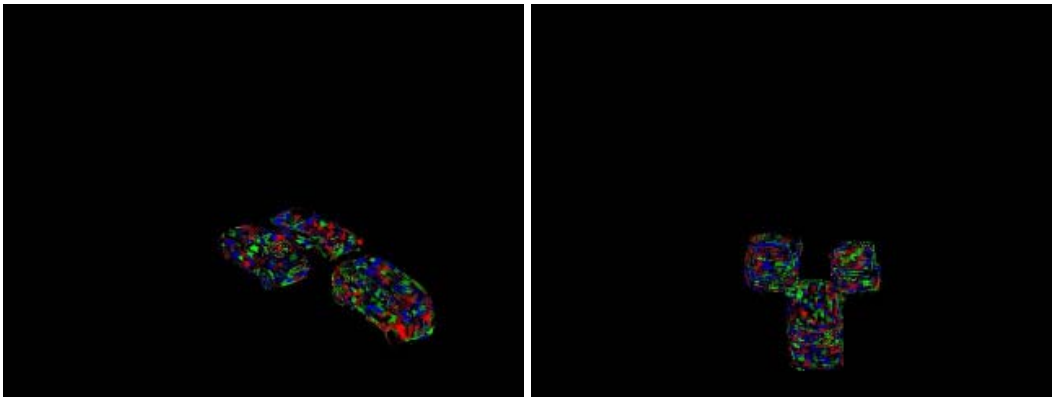
Execution Time (sec.)	**320 x 240**		**640 x 480**	
Number of Segments	**RSST**	**K-Means**	**RSST**	**K-Means**
300	192	0.9	234	11.2
500	210	0.93	295	13
1000	272	1.9	326	17.5



(a) **(b)**

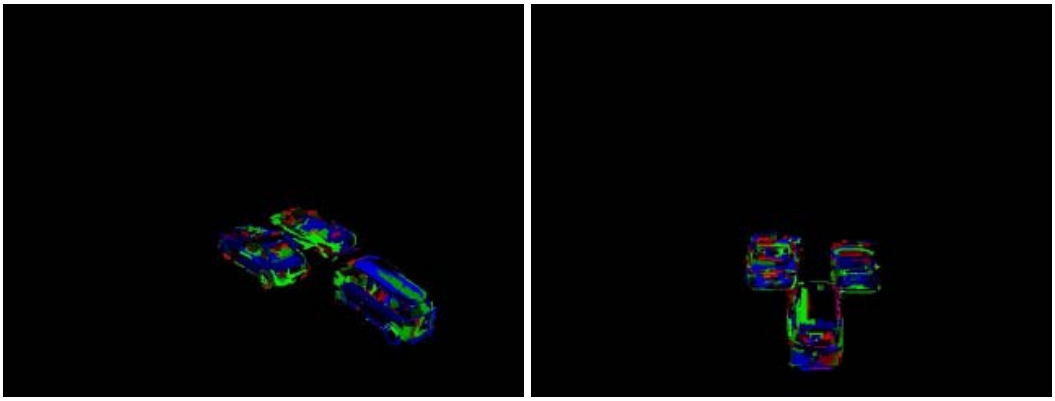
(c) **(d)**

Figure 5.2 Input image (a) First camera and (b) second camera view. (c) and (d) are corresponding foreground masks.



(a) **(b)**

Figure 5.3 K-means segmentation method results with K=1000 segments; (a) first camera and (b) second camera view.



(a) **(b)**

Figure 5.4 RSST segmentation method results with 1000 segments; (a) first camera and (b) second camera view.

It could be concluded that RSST segmentation method has a heavy computational load and hence, it is not suitable for real-time applications. At the end of the experiments both methods have similar results subjectively in terms of their

segmentation performance and K-means segmentation method is accepted to be more appropriate due to its low computational load.

5.4 Point Transfer Using Trifocal Tensor

Camera point of view and the angle between the camera and the object plane are the main reasons for the occlusions. It is obvious that the most appropriate view in which the occlusion is minimized is the top-view for the vehicles and people that move on a plane (note the possible shortcomings of top-view for the flying vehicles). Hence, in the proposed method, a top-view is generated to handle occluded objects and this step is performed by using trifocal tensor where the camera calibration is required.

5.4.1 Camera Calibration

The camera matrix P is computed by using point correspondences $X_i \leftrightarrow x_i$ between 3D feature points and their projections onto 2D image points. These feature points must be distinguishable and salient elements of an image with known 3D coordinates. For this purpose 3D patterns with known structures could be used. Checker-board structures, with precisely known dimensions as shown in Figure 5.5, are used to calibrate cameras.

5.4.1.1 Feature Point Detection

There are various feature detection algorithms in literature [45][46][47][48][49] [50]. In the past, Harris and Stephens [46] have improved the Moravec corner detector [51] by performing an analytic expansion about the shift origin, instead of discrete shifts. Moreover, robustness to noise was also improved by using a smooth circular window.

Figure 5.5 Checker-board patterns which are used to calibrate cameras

Later, the method by Lowe [50] extracts some distinctive features from the images by using difference of Gaussians. These features, denoted as SIFT, are invariant to scale changes, as well as rotations, and robust to illumination differences and camera movements. It has been shown that Harris corner has more consistent results when compared to many other methods [46]. Therefore, Harris corner is utilized to find features from the calibration patterns.

5.4.1.1.1 Harris Corner

Moravec's corner detector [51] functions by considering a local window in the image, and determining the average changes of image intensity that result from shifting the window by a small amount in various directions. These shifts result in

- small changes, if the windowed image patch has *constant intensity* values
- small changes in the direction of edges and large changes in the direction perpendicular to the *edges*
- large changes in all directions for *corners*.

The change in intensity C produced by a shift $(\Delta x, \Delta y)$ for point (x,y) is formulated as

$$C(x, y) = \sum_{w} | I(x, y) - I(x + \Delta x, y + \Delta y) |^2 \tag{5.1}$$

where I is the image intensity and w is the image window centered at (x,y). Moravec's corner detector simply looks for local maxima in min{C} above some threshold value. Harris corner detector [46] improves the anisotropic response of Moravec's corner detector (at every 45 degrees) by approximating the shifted image by a Taylor expansion truncated to the first order terms. Then

$$I(x_i + \Delta x, y_i + \Delta y) \approx I(x_i, y_i) + \begin{bmatrix} I_x(x_i, y_i) & I_y(x_i, y_i) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{5.2}$$

in which the first gradients are approximated by

$I_x = I \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \partial I / \partial x$ and $I_y = I \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T = \partial I / \partial y$.

Then equation (5.1) becomes

$$C(x_i, y_i) = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} \sum_w (I_x(x_i, y_i))^2 & \sum_w I_x(x_i, y_i) I_y(x_i, y_i) \\ \sum_w I_x(x_i, y_i) I_y(x_i, y_i) & \sum_w (I_y(x_i, y_i))^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$= \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M(x_i, y_i) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{5.3}$$

Also noisy response of Moravec's corner detector is improved by using a Gaussian window.

$$w_{x,y} = \exp\left(-(x^2 + y^2)/2\sigma^2\right) \tag{5.4}$$

C(x,y) is closely related to local autocorrelation function, with M describing its shape at the origin. Let λ_1 and λ_2 be the eigenvalues of M. Then the eigenvalues form a rotationally invariant description. There are three cases to be considered as in the Moravec's detector:

1. If both λ_1 and λ_2 are small, so that the local auto-correlation function is flat (i.e., small change in $M(x, y)$ in any direction), the windowed image region is of approximately constant intensity.

2. If one eigenvalue is high and the other low, so the local auto-correlation function is ridge shaped, then only local shifts in one direction (along the ridge) cause little change in $C(x, y)$ and significant change in the orthogonal direction; this indicates an edge.

3. If both eigenvalues are high, so the local auto-correlation function is sharply peaked, then shifts in any direction will result in a significant increase; this indicates a corner.

Therefore, a function R is needed to represent the measure of corner response which is required to be a function of λ_1 and λ_2. Trace and determinant operations are suitable for formulation, as this avoids the explicit eigenvalue decomposition of M.

Thus, if $M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$ then $Tr(M) = \lambda_1 + \lambda_2 = A + B$ and $Det(M) = \lambda_1 \lambda_2 = AB - C^2$.

Finally,

$$R(M) = Det(M) - kTr(M)^2 \tag{5.5}$$

The features, which are detected by using Harris corner method in sub-pixel accuracy, are shown in Figure 5.6.

Figure 5.6 Features that are detected by using Harris corner method

5.4.2 Point Transfer

Camera projection matrix of the imaginary top-view camera is obtained by assuming that the camera is located above the center of region of interest without any rotation and translation, and with zero principal point offset. Moreover, an appropriate *f* (focal length) of the top view camera is chosen by considering the resolution of the image that will be generated and the area of the region of interest.

Epipolar match of each segment in the first camera view must be determined for the second camera view in order to be able to transfer them onto the top-view. Therefore, the color models (color histograms) of all the segments (both in the first ad second views) are generated. Then, for segment's center point p_i (center of mass) in the first camera view, corresponding epipolar line l_i' is obtained by using fundamental matrix F_{12}. Epipolar match of the p_i, which is p_i', must lie on the epipolar line l_i'. For finding the match of p_i, its color model is compared with all segments in the second camera view which have intersection with the epipolar line l_i'. A distance measure is used to compare the color models, and the segment with minimum distance is selected as the best match.

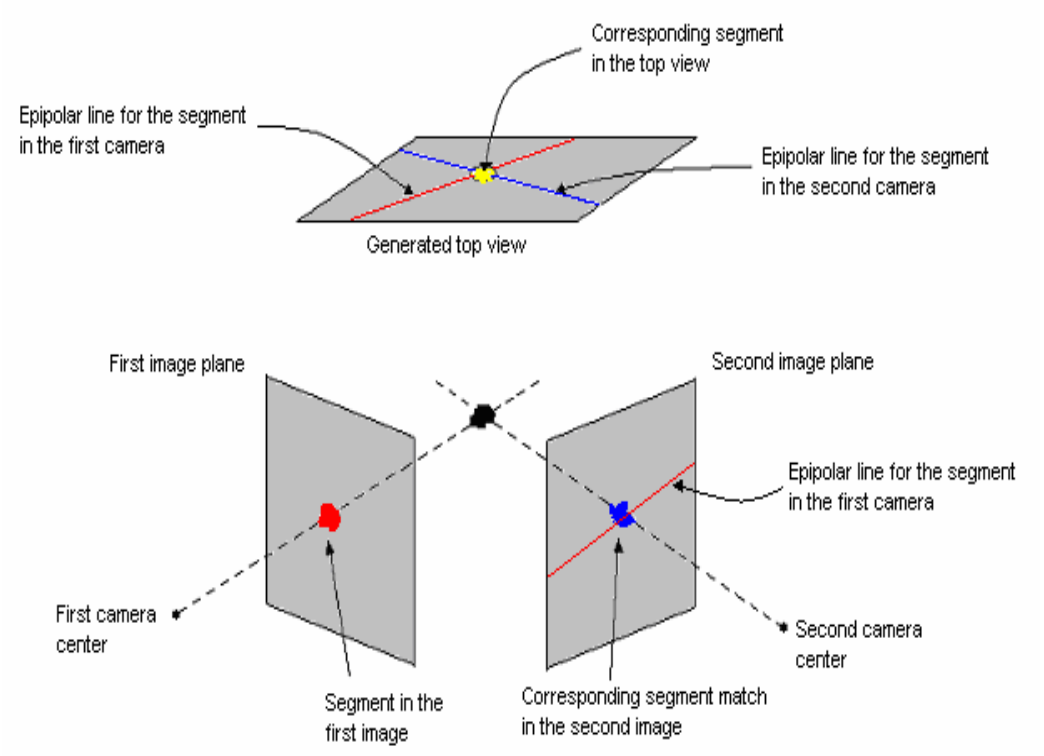
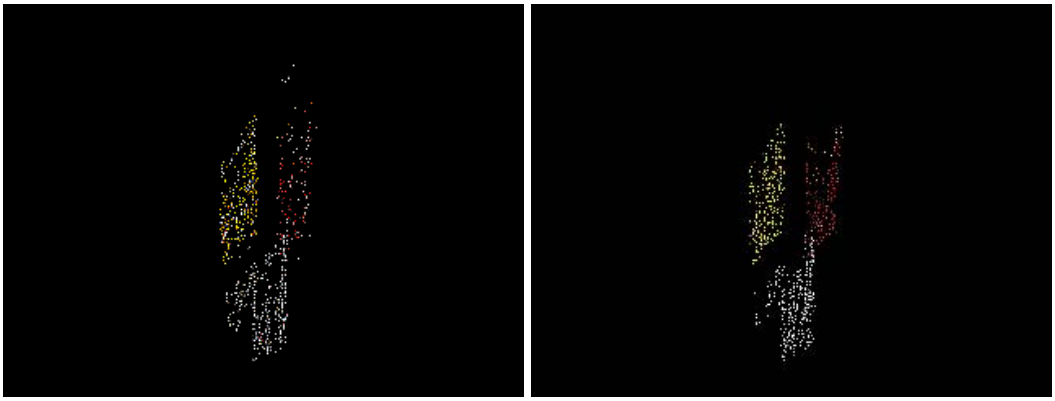


Figure 5.7 Point transfer by using trifocal tensor

Then for matched point pair p_i and p_i', corresponding two epipolar lines on the top-view is determined by using F_{13}.and F_{23}. Since the required point p_i'' must lie on both of the epipolar lines, taking the intersection of the epipolar lines results in the required point p_i''.

The discussed procedure, which is also illustrated in the Figure 5.Figure 5.7, is repeated for all segments in the first camera view and a top-view that is consisting of points, which correspond to segment centers, is generated.

During simulations, Kullback-Leibler divergence [75] is used as a distance measure to compare the color models of segments. The images are segmented into 1000 parts and shown in the Figure 5.3 and Figure 5.4. The results for point transfer are shown in Figure 5.8.



(a) **(b)**

Figure 5.8 Generated top-views by using images segmented with (a) RSST and (b) K-means

5.5 Graph-Based Clustering

Generated top-view is composed of points which are obtained independent of each other. These points must be segmented to cluster them into individual objects, and the only information which is available is positions and colors of points. Point clustering is performed by using 'minimal spanning tree' which is discussed in Chapter 3.

5.5.1 Constructing Minimal Spanning Tree

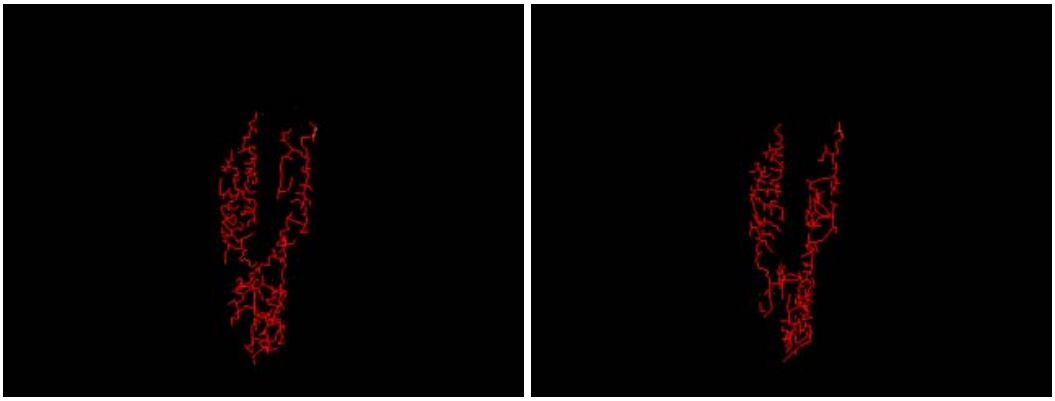
Top-view points, obtained by intersecting the epipolar lines of the segment pairs, are defined as nodes or vertices of the undirected graph. In order to decrease the computational cost, only one node is chosen in a $N \times N$ neighborhood of that node. Assuming that the segments of the same object are close to each other and most of the cases they have similar color values, the weights of the links or edges between these nodes could be defined as follows;

Weight value of the edge between nodes-*i* and *j* is

$$W_{ij} = \alpha H_{diff} + \beta D_{diff} \tag{5.6}$$

where H_{diff} is the difference between the hue values of the nodes-*i* and *j*, D_{diff} is the Euclidian distance between nodes-*i* and *j*, whereas α and β are the weights between these measures. By using the weight function in (5.6), points, which have similar color values and close to each other, are connected by edges with smaller weight values.

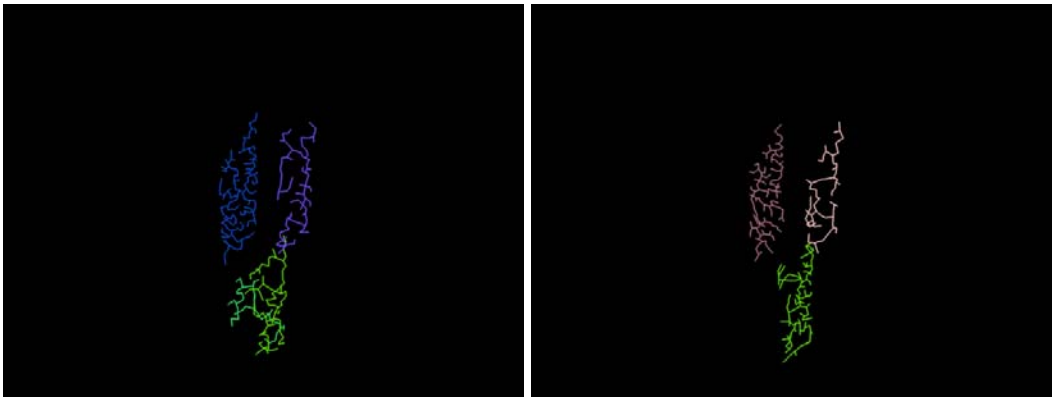
Next, Kruskal's algorithm [26] is used to generate minimal spanning tree of the undirected graph, which is shown in Figure 5.9.



(a) **(b)**

Figure 5.9 Generated minimal spanning trees by using the images segmented with (a) RSST and (b) K-means

Edges, whose weights are greater than certain threshold value in the minimal spanning tree, are cut and minimal spanning forests are generated. Then, the forests, which have nodes smaller than certain threshold, are qmasked to eliminate noisy and uncertain regions. The resulting forests are labeled with different colors and the individual objects are generated, as shown in Figure 5.10.



(a) **(b)**

Figure 5.10 Generated individual objects by using the images segmented with (a) RSST and (b) K-means

As a result, a single foreground mask is divided into three regions which are corresponding to different objects (vehicles).

5.6 Experimental Results and Conclusion

The method discussed to segment occluded objects is tested by using 14 data sets. The performance of the overall system is obtained as quite convenient for segmenting partially occluded objects. However, under strong occlusions, epipolar matching cannot be performed accurately and the generated top view does not represent the foreground objects. Moreover, the foreground mask is under-segmented, when the objects in the mask have similar color values and they are close to each other, since the transferred points are not distinguishable. Typical two experimental results are given in Figure 5.11 and Figure 5.12.. False epipolar matching results also decrease the performance of the method. However, as the number of segments increase, these false epipolar matches are compensated and satisfactory results could be obtained.

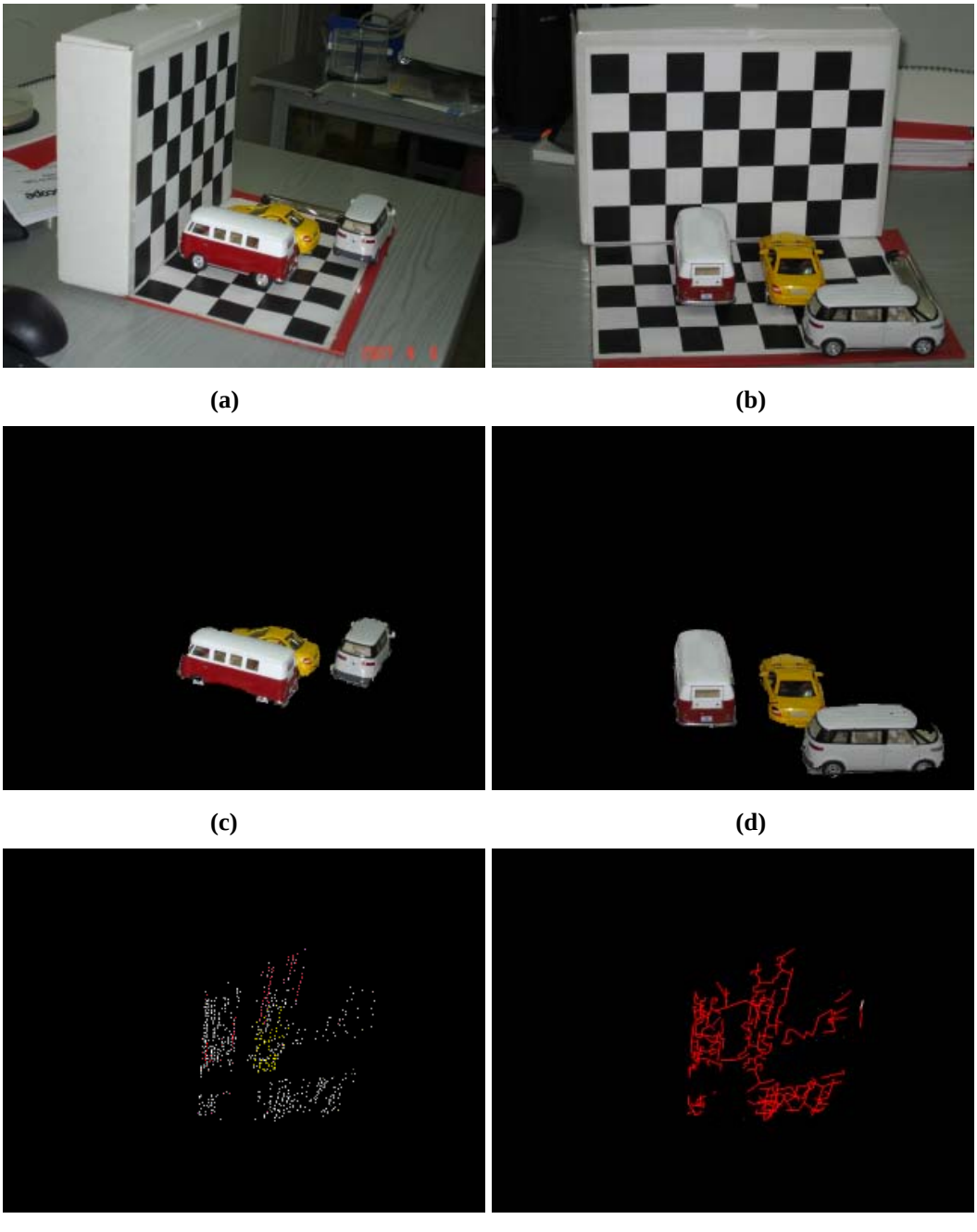


Figure 5.11 Simulation results of the dataset 7, (a) and (b) are the input images, (c) and (d) are the corresponding segmented foreground masks, (e) is the calculated top-view points, (f) is the constructed minimal spanning

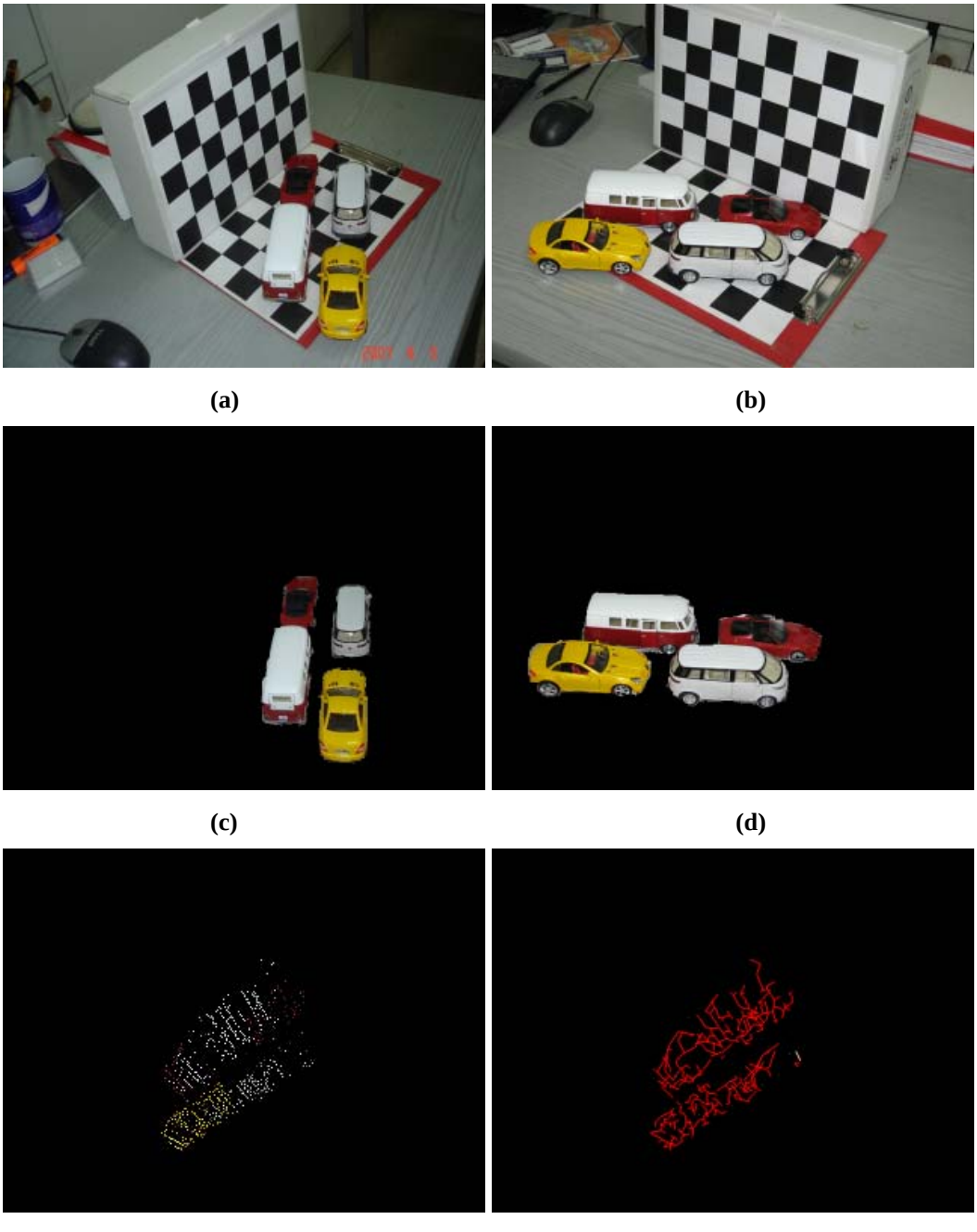


Figure 5.12 Simulation results of the dataset 6, (a) and (b) are the input images, (c) and (d) are the corresponding segmented foreground masks, (e) is the calculated top-view points, (f) is the constructed minimal spanning

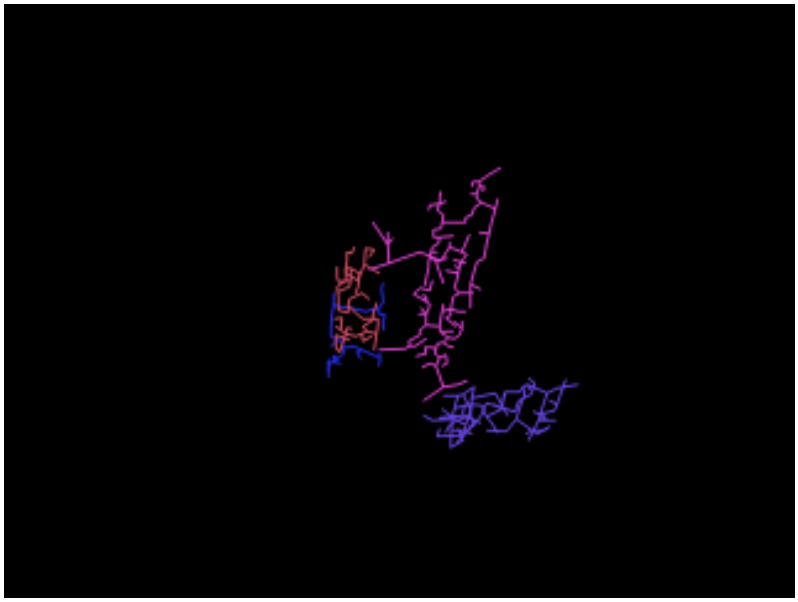


Figure 5.13 Simulation results of the dataset 7, clustered minimal spanning forests

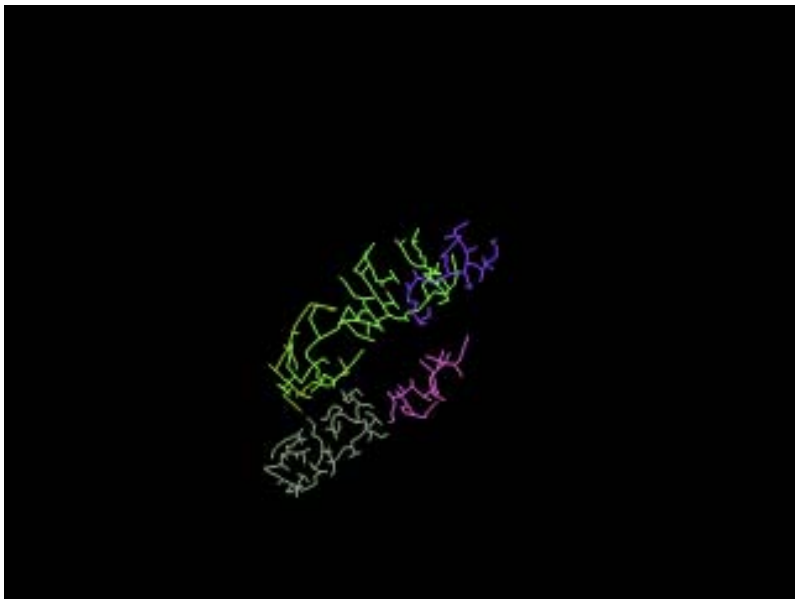


Figure 5.14 Simulation results of the dataset 6, clustered minimal spanning forests

CHAPTER 6

MULTI-CAMERA TRACKING AND EVENT RECOGNITION

Tracking of the objects in the observed scene is another vital property in video surveillance systems. One common cause of tracking failure in single-camera configuration is due to the static and dynamic occlusions. When occlusion is minimal, single camera is sufficient for tracking. However, as the density of static and dynamic occlusions is increased, objects are segmented in an erroneous way or become totally lost; hence, lack of visibility results in tracking failure. Since the multi-camera configurations have larger field of view due to observation from different angles, it is expected that these systems should be capable of resolving static, as well as dynamic occlusions better than single-camera configurations. It should be noted that in multi-camera systems, occlusions might occur in different time instants at separate views, and the overall system should be able to track occluded objects successfully after exploiting the available track information at different views.

The use of multiple cameras might lead to extraction of more reliable trajectories due to the utilization of 3D information, compared to a single camera. By the help of proposed multi-camera tracking method, obtained trajectories of the object can be used during event recognition. Since occlusion-free 3D information is being

utilized, these trajectories lead to better interpretations of activity type that is of significant interest.

In this chapter, a tracking method by using multi-camera (2 cameras) system is explored. First, various related work in the literature is briefly discussed. Then, performed tracking algorithm and trajectory generation method are proposed. Finally, event recognition by using multi-view trajectories is discussed. Simulation results and conclusion are given at the end of the chapter.

6.1 Related Work

There are various methods in multi-camera tracking [53]. The main idea of multi-camera employment is fusing 2D information due to the cameras in order to obtain 3D information for better occlusion handling and continuous tracking. Kettnaker et al. [43] perform motion detection and tracking on each video stream, and then, synthesize the tracking results of different cameras to obtain an integrated trajectory. The major drawback of this system is the requirement a set of allowable paths, and a set of transition probabilities and times, to be given as input.

Dockstader et al. [55] propose a distributed system for tracking multiple interacting persons which provide increased robustness against temporary feature point occlusions. A Bayesian network is employed to fuse 2D state vectors acquired from various image sequences to obtain a 3D state vector. However, motion extraction is a difficult problem to solve in real time, and the complexity of this algorithm is quite high.

Collins et al. [56] propose an algorithm that obtains an integrated representation of an entire scene by fusing information from every camera into a 3D geometric coordinate system. However, wide-baseline triangulation for 3D localization is

still a difficult process for cluttered scenes with densely located objects and significant occlusion.

Chang and Gong [59][60] propose a method to track people by using two camera views in an indoor area. The system is based on fusion of Bayesian modalities, such as geometric and appearance modalities for cooperative tracking. In order to track individuals seamlessly, the system assigns an identity to a detected subject and keeps tracking the subject with this identity. If this subject has already appeared in other cameras or loses its identity during tracking, the system then passes identity and re-assigns it to this subject by matching subjects across other camera views by using a Bayesian framework. However, their system is robust indoors, since the lighting conditions are fairly stable, whereas this approach could only be applied outdoors during times when the lighting conditions do not vary considerably.

Kogut and Trivedi [61] propose a system for traffic surveillance. They propose a hybrid method, utilizing color and spatial information to construct labeled graphs of small groups, or platoons, of cars, as they travel through a camera site. Then, the cars are matched by using probabilistic graph matching between the camera sites. Therefore, the system could track platoons of vehicles between each camera site. However, they do not propose a solution for the occlusion scenarios and they tried to minimize the errors due to occlusions by using the high perspective view. However, high perspective view is not available in most of the traffic surveillance systems due to the camera placements.

Khan et al. [62][63][64][65][66] present a system for multi-view surveillance that can be applied to both indoor and outdoor environments by a set of uncalibrated cameras. Their method can automatically identify the field of view boundaries between overlapping cameras. Once this information is available, it is possible for the multi-view tracking algorithm to consistently assign the correct identity to objects, even when they appear in more than one camera view. In order to handle

the scenario of tracking objects between non-overlapping cameras, they use a combination of spatio-temporal information and color cues. They assume that training data is available for objects moving between the fields of view of each non-overlapping camera.

Black et al. [57][58] present a multi-view tracking method using a set of calibrated cameras. They perform background subtraction and use homography to map segmented objects in different views. 3D line intersection algorithm is used to find 3D coordinates of matched objects. The Kalman filter is used to track each object in 3D world coordinates and 2D image coordinates. This interesting method has quite acceptable experimental results and seems to have real-time performance due to the simplicity of the proposed algorithm.

6.2 Tracking and Event Recognition from Multi-view Video

The previous research efforts show that the multi-camera employment is expected to give promising results during tracking. Especially for the occluded scenes, collaboration of multi-camera configurations leads better handling of track losses. In this thesis, the approach by Black et al. [57] is pursued for developing a tracking algorithm due to their promising results with their relatively simple algorithm.

As stated in [57], moving object segmentation is performed by background subtraction and Kalman filter is used to track segmented objects in each of the views. When there is no measurement associated with the track due to occlusion, other views are used to generate measurement. In order to fuse the information coming from different cameras, a relation between these two tracks should be defined. The simplest way of relating different views is point transferring via homography, since epipolar matching, by using epipolar lines, is computationally expensive and difficult to perform for wide-baseline configurations.

In the upcoming sections of this chapter, Kalman filter paradigm is briefly discussed in Section 6.3 to better understand any tracker, and the tracking algorithm is explained in Section 6.4. After tracking, motion models of the objects must be extracted to classify their behaviors. Therefore, in each camera view, from the extracted 2D trajectories of the objects, multi-view trajectories are obtained. Then, Hidden Markov Models [22][23] are used to classify the extracted 2D trajectories of the tracked object and event recognition is performed according to this classification. In Section 6.5, event recognition via HMMs is explained in detail.

6.3 Kalman Filter

A Kalman filter [20][21], is used to estimate the state X of a discrete-time controlled process that is governed by the linear stochastic difference equation, as

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{6.1}$$

with a measurement

$$z_k = Hx_k + v_k \tag{6.2}$$

w_k and v_k are process and measurement noise, which are independent of each other, white and usually assumed to have normal probability distribution.

$p(w) \sim N(0, Q)$ where Q is the process noise covariance

$p(v) \sim N(0, R)$ where R is the measurement noise covariance

The Kalman filter estimates a process by using a form of feedback control. The filter estimates the process state at a certain time and then obtains feedback in the

form of noisy measurements. The equations for the Kalman filter fall into two groups

- Time update equations: obtain a priori estimate for next time step
- Measurement update equations: incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

Next, time update equations act like predictors, while the measurement update equations are playing the role of a corrector. These equations are as follows:

Time update equations: (Kalman predict)

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement update equations: (Kalman Correct)

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

where P_k is the estimate error covariance.

6.4 Proposed Multi-camera Tracking

In the proposed approach, tracking is performed in both of the views and trackers in different views are related to each other via homography. The object states are tracked in 2D by using separate Kalman filters. The object state in 2D Kalman filter includes the image location of object as well as its velocity in pixels. Moreover, constant speed assumption for object velocity is used.

Utilized models are as follows;

2D state model:

$$X_i = [x \quad y \quad v_x \quad v_y] \tag{6.3}$$

State transition model:

$$A_i = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.4}$$

Observation model:

$$O_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{6.5}$$

When an object is observed for the first time, a separate Kalman filter is initiated for this object. Then, for next incoming frame, background subtraction is performed and a set of foreground moving objects are obtained. Among these objects, the nearest moving object to the predicted state of the tracked object is labeled as the next position of that object and the position is used to update the corresponding Kalman filter. Hence, object association between the consecutive frames is achieved via Kalman predictions. If the distance between the nearest moving object and the predicted state of tracked object is larger than some certain threshold, or there is no moving object in the foreground mask, then this situation is named as “no measurement case”. When “no measurement case” occurs in one of the views, mostly due to occlusions, while there is a measurement in the other view for tracked object, the position information projected from the other view is used as a measurement to update for the corresponding Kalman filter. In order to

utilize this information, a match of the object in the other camera view must be determined. Given a set of detected moving objects in each camera view, a match between a correspondence pair is defined when the following transfer error condition is satisfied:

$$(x' - Hx)^2 + (x - H^{-1}x')^2 < \tau \tag{6.6}$$

where x and x' are image coordinates in the first and second camera views, respectively. This constraint is applied to determine correspondence between the moving objects, detected in each camera view. The representative coordinate of an object is assumed to, be the closest point of its foreground mask to the ground plane (which is generally the rear end of the mask), since its location minimizes the projection errors. When "no measurement case" occurs in both of the views associated with the tracked object, updates of the Kalman filters are calculated separately by using corresponding predicted state values.

The aforementioned steps are repeated until the tracked object leaves the field of view (FOV). When the object leaves the common FOV, its trajectories for both of the cameras are extracted for event recognition. The proposed method is illustrated in Figure 6.1. In the first frames from camera 1 and 2, 3 objects are observed for the first time in the common FOV and observed objects in different views are matched to each other ($a - a'$, $b - b'$ and $c - c'$). Also, separate Kalman filters are initiated for these objects and next-states are predicted. In the second frames, there is no measurement for $Object - a$ and the position information from second view, $Object - a'$, is projected as a measurement to update corresponding Kalman filter. In the third frames, there is no measurement for $Object - c'$ and the measurement from first view, $Object - c$, is projected instead. In the final image pair, 'no measurement case' occurs for both of the $Object - b$ and $Object - b'$. Therefore, Kalman filter updates are performed by using previously predicted Kalman states, separately.

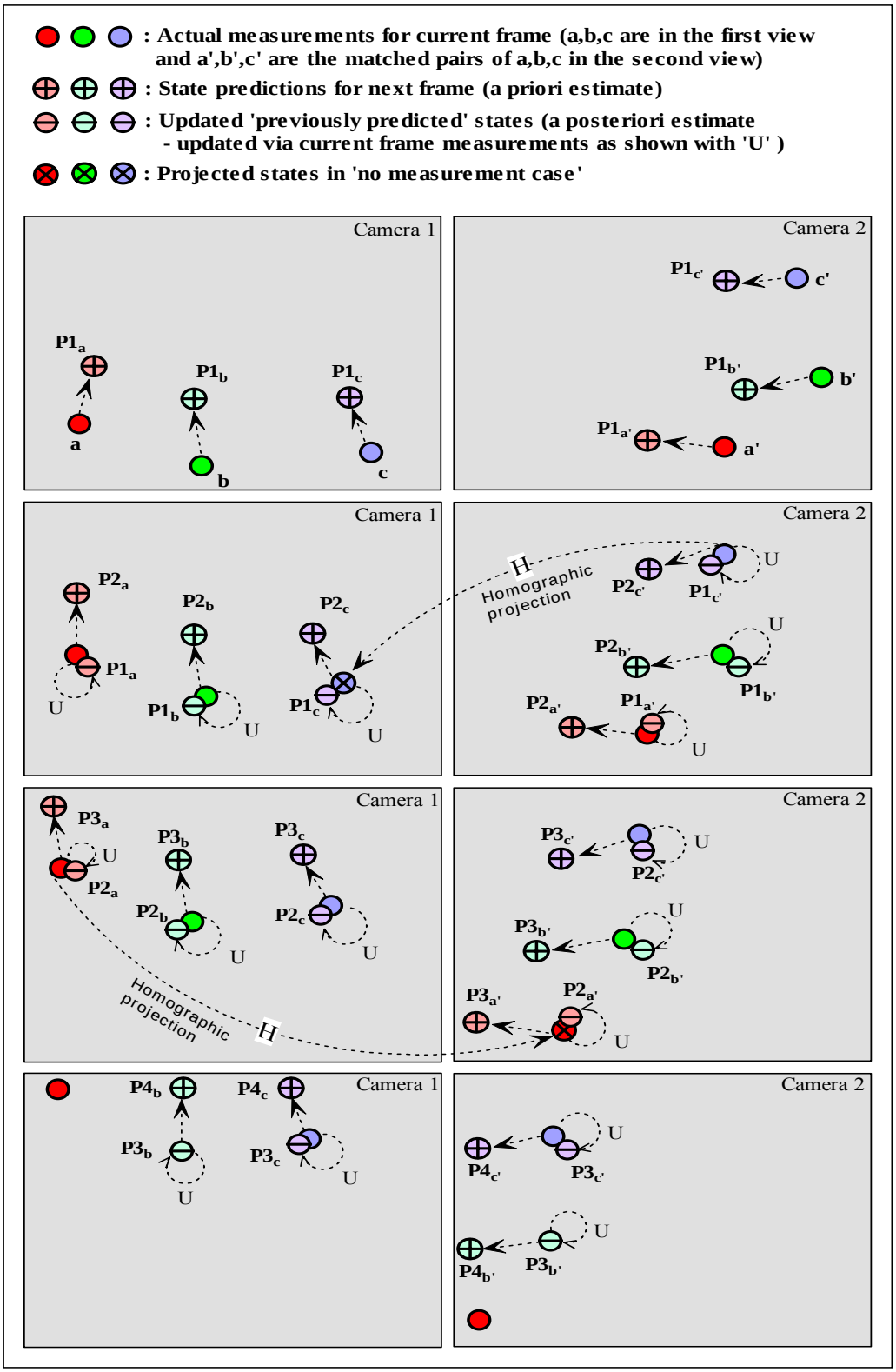


Figure 6.1 Multi-camera tracking for consecutive time instants

6.5 Multi-view Event Recognition

A similar method, which is already discussed in Chapter 2, is utilized to recognize the spatio-temporal events that occur in the observed scene. The events are classified into two sets, as 'normal' and 'abnormal' due to their spatio-temporal behaviors that are stored in their trajectories. Normality is simply defined by being in the training set. The extracted object trajectories, which are accepted to be 'normal', are used to train a GM-HMM and new trajectories are classified by using this model. More information about GM-HMMs can be found in Chapter 2.

For the single camera case, the measurements can be obtained by using only one sensor. Therefore, the extracted data are limited, compared to the multi-camera case. In the proposed multi-camera (two cameras) event recognition method, three different trajectory vectors are defined for each object, such that the first one is the image locations of the object in the first camera view, whereas the second one is the image locations of its matched pair in the second camera view. The last trajectory vector is obtained by concatenating the first and the second trajectories.

The positions of the i^{th} object in the k^{th} frame of first and second camera are shown as the position vectors;

$$p_{k_1}^{i} = [x_{k_1}^{i} \quad y_{k_1}^{i}] \text{ and } p_{k_2}^{i} = [x_{k_2}^{i} \quad y_{k_2}^{i}] \tag{6.6}$$

where x and y are the image coordinates in the corresponding camera views.

Then, the first and second trajectory vectors consist of

$$tr^1(i) = \begin{bmatrix} x^i_{m_1} & y^i_{m_1} \\ x^i_{m_1+1} & y^i_{m_1+1} \\ . & . \\ . & . \\ x^i_{n_1} & y^i_{n_1} \end{bmatrix} \text{ and } tr^2(i) = \begin{bmatrix} x^i_{m_2} & y^i_{m_2} \\ x^i_{m_2+1} & y^i_{m_2+1} \\ . & . \\ . & . \\ x^i_{n_2} & y^i_{n_2} \end{bmatrix} \tag{6.7}$$

The third trajectory vector is

$$tr^3(i) = \begin{bmatrix} x^i_{m_1} & x^i_{m_2} & y^i_{m_1} & y^i_{m_2} \\ x^i_{m_1+1} & x^i_{m_2+1} & y^i_{m_1+1} & y^i_{m_2+1} \\ . & . & . & . \\ . & . & . & . \\ x^i_{n_1} & x^i_{n_2} & y^i_{n_1} & y^i_{n_2} \end{bmatrix} \tag{6.8}$$

where *m* denotes the starting frame number, in which the object enters the FOV, and *n* is the end frame number, in which the object leaves the FOV.

These three trajectory vector types are used to train 3 different GM-HMMs and the incoming trajectory vectors are tested separately by using these three models in order to check their applicability to these trained models. The simulation results are given in the Section 6.6.2.

6.6 Simulation Results and Conclusion

In this section, the simulation results for multi-view tracking and event recognition are presented and discussed. The discussed two methods are tested by using various video sequences which are in MPEG-2 format recorded at 30 fps

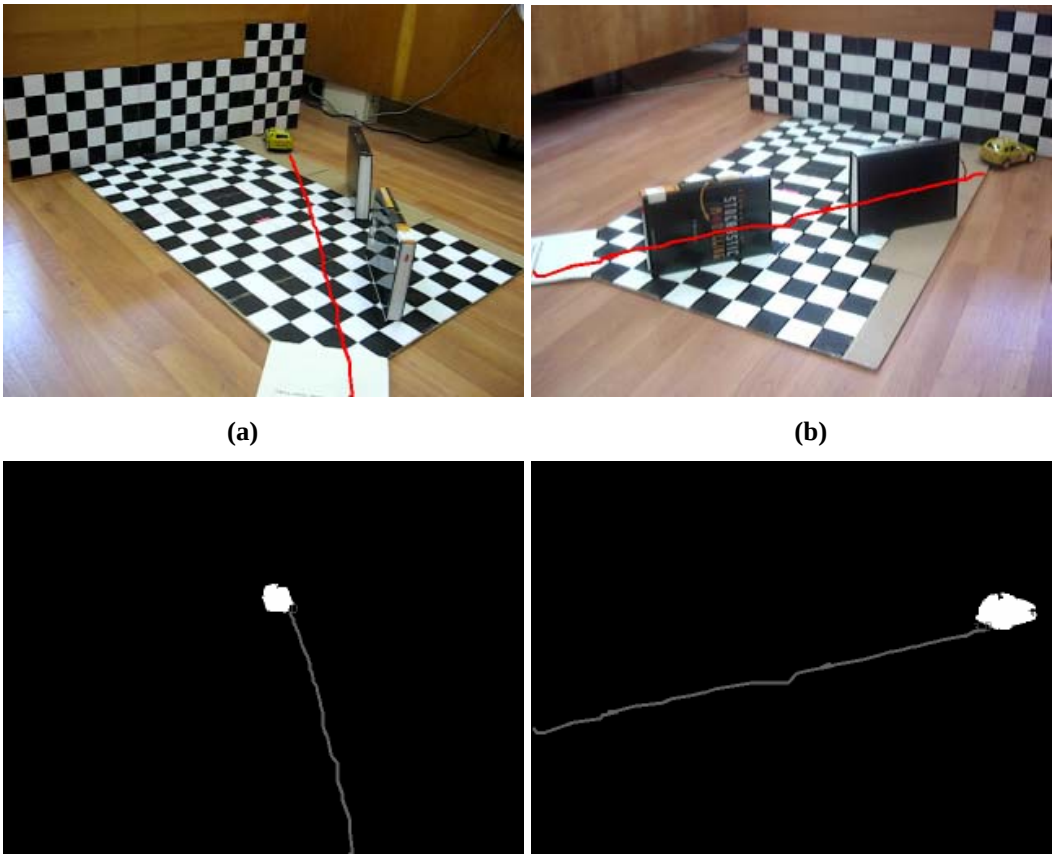
with a resolution of 640x480. All these simulations are conducted in controlled laboratory environments.

6.6.1 Simulation Results for Multi-view Tracking

The discussed method is tested by using 14 different video sequences each of which a vehicle runs in front of a camera and 2 typical results are given in the Figure 6.2 and Figure 6.5. Also, individual performances of trackers (without other view measurement assistance) are tested by using the same video sequences and the results are given in the Figure 6.3 and Figure 6.6. The generated top-view trajectories for both cases are presented in the Figure 6.4 and Figure 6.7.

The performance of the proposed method is strongly dependent on the correct matching performance of the tracked object between the two views. Therefore, the object must be segmented correctly in the initial frames, when it enters the common FOV. Moreover, a couple of correct measurements are needed to initiate the Kalman filters. The small variations in the object trajectories are caused by erroneous foreground masks. During background subtraction step, the foreground mask of the object differs slightly in size (especially along its borders) between consecutive frames. Therefore, the location of the object, which is the rear-end point of its foreground mask, slightly vibrates between frames.

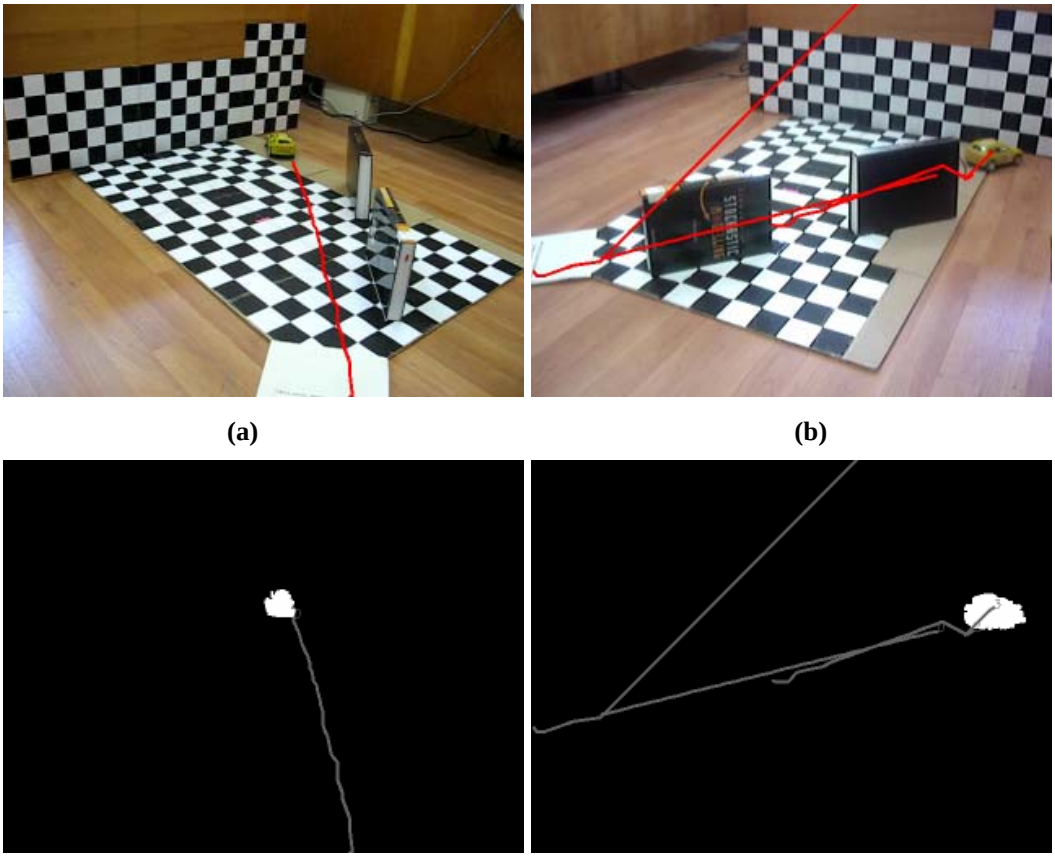
The proposed method has an obvious advantage compared to single-camera tracking, when tracked object stops behind an obstacle. As long as one of the cameras continues to observe the object, it can be tracked correctly along the frames. However, in the single-camera case, Kalman tracker for the occluded object would fail and object becomes lost. As shown in the simulation results, Kalman trackers (without assistance) failed and lost the track of object when it passed behind an obstacle. Also new Kalman trackers are mistakenly initiated since lost objects are incorrectly identified as appearing for the first time.



(a) **(b)**

(c) **(d)**

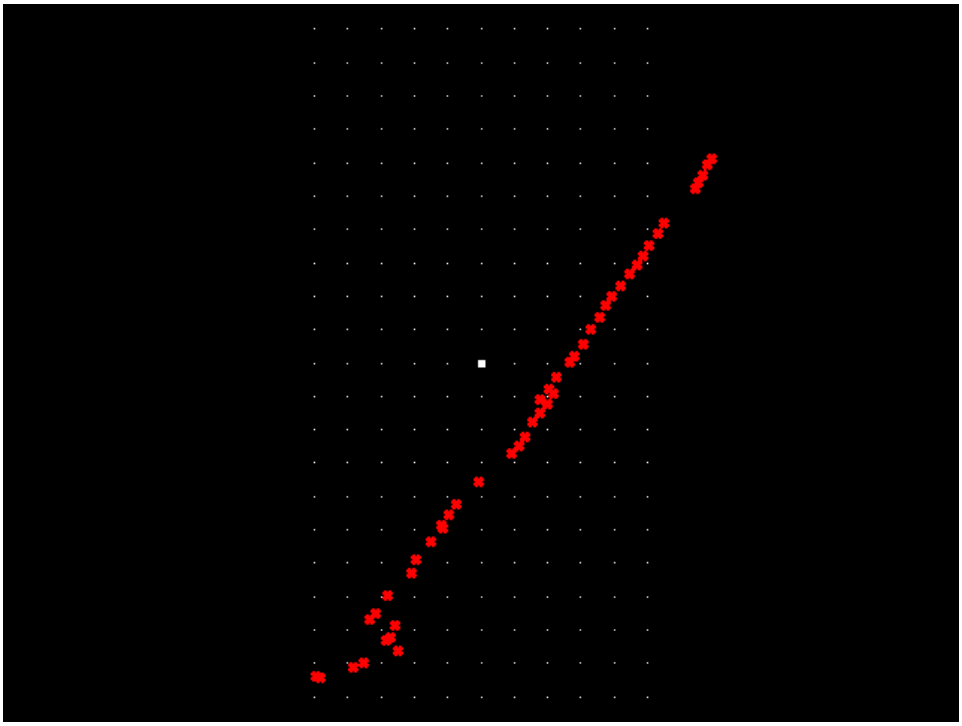
Figure 6.2 Simulation results of multi-camera tracking algorithm tested on dataset 1 (a) and (b) are corresponding camera views, (c) and (d) are the trajectories of segmented object in corresponding views



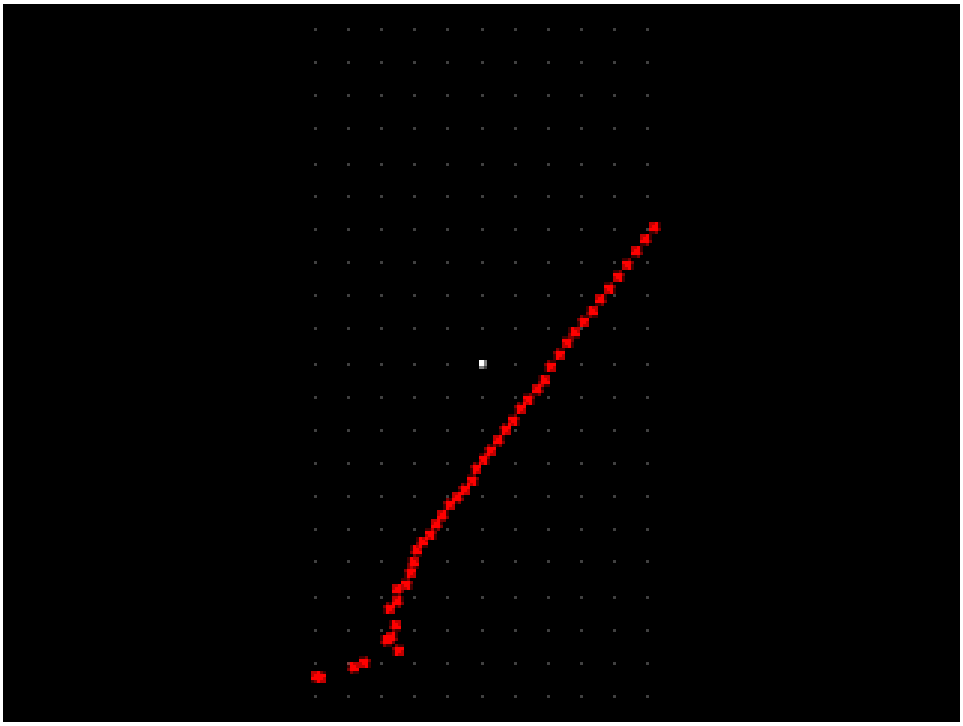
(a) **(b)**

(c) **(d)**

Figure 6.3 Simulation results of multi-camera tracking algorithm, in which other view measurements are not used in 'no measurement case', tested on separate views of dataset 1 (a) and (b) are corresponding camera views, (c) and (d) are the trajectories of segmented object in corresponding views

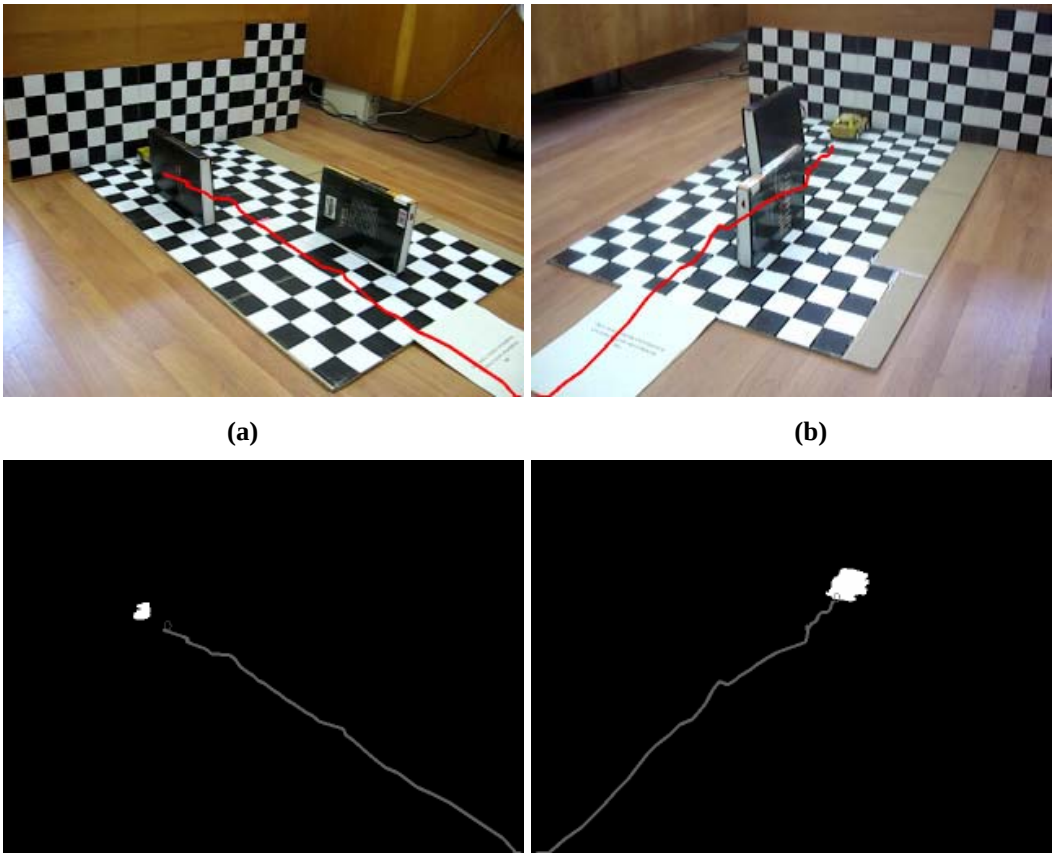


(a)



(b)

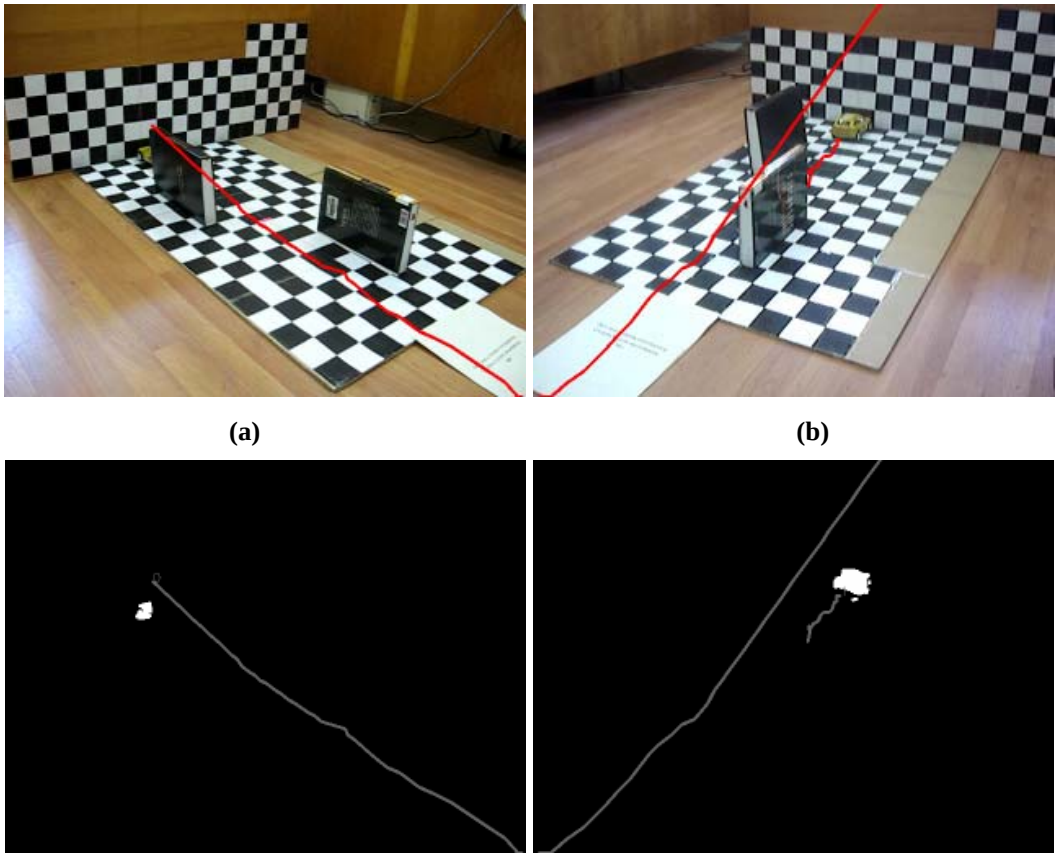
Figure 6.4 Generated top-view trajectories by using extracted trajectories from dataset 1: in 'no measurement case' other view measurements are (a) used (b) not used



(a) **(b)**

(c) **(d)**

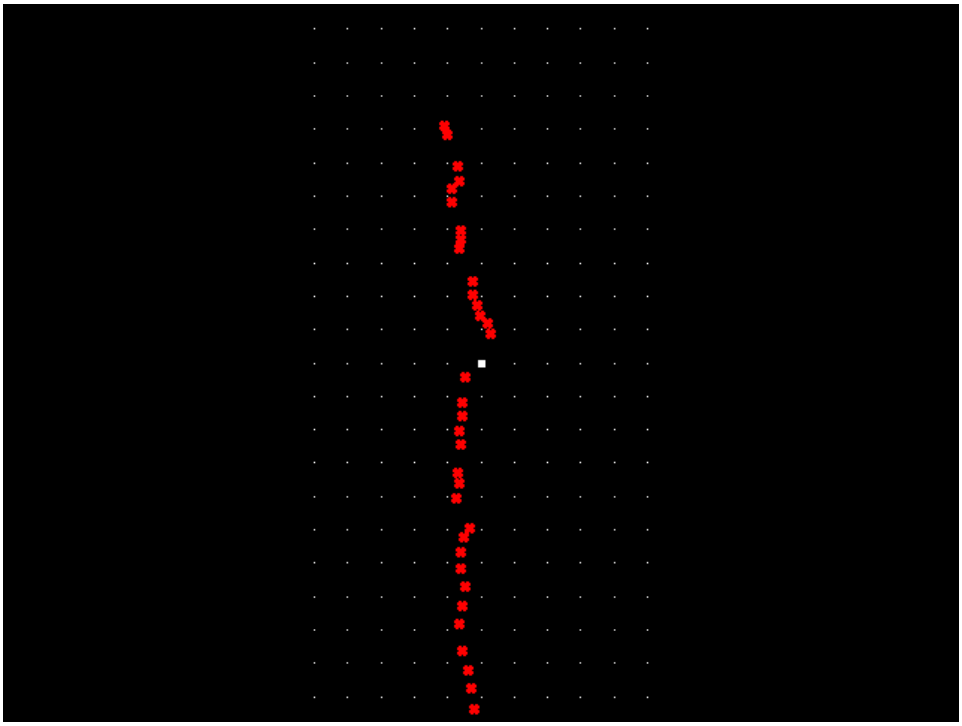
Figure 6.5 Simulation results of multi-camera tracking algorithm tested on dataset 6 (a) and (b) are corresponding camera views, (c) and (d) are the trajectories of segmented object in corresponding views



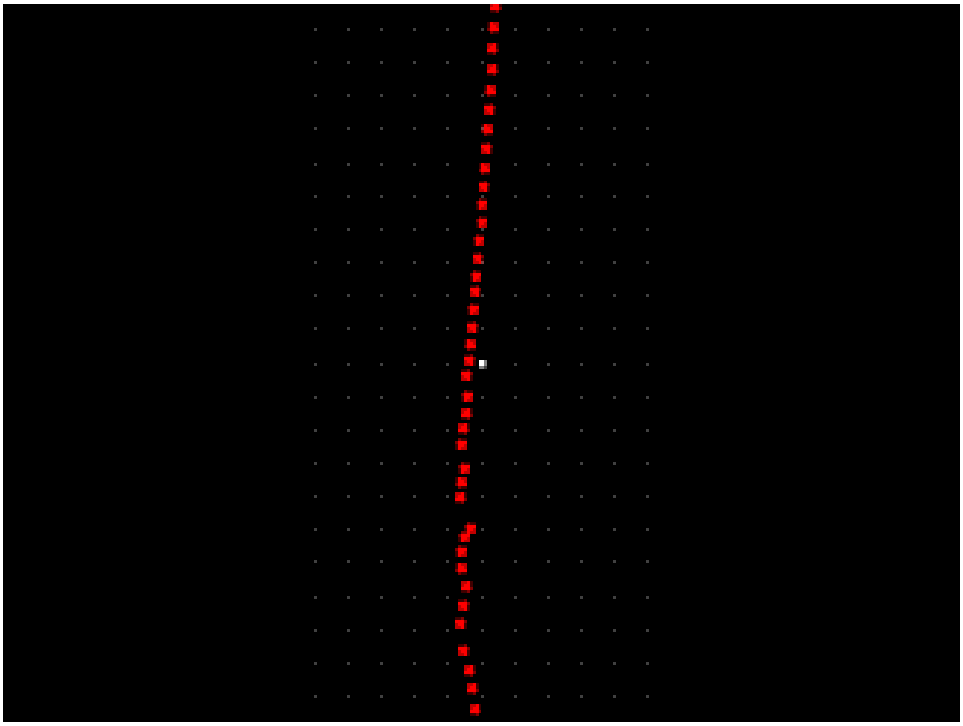
(a) **(b)**

(c) **(d)**

Figure 6.6 Simulation results of multi-camera tracking algorithm, in which other view measurements are not used in 'no measurement case', tested on separate views of dataset 6 (a) and (b) are corresponding camera views, (c) and (d) are the trajectories of segmented object in corresponding views



(a)



(b)

Figure 6.7 Generated top-view trajectories by using extracted trajectories from dataset 6: in 'no measurement case' other view measurements are (a) used (b) not used

6.6.2 Simulation Results for Multi-view Event Recognition

Training trajectory vectors that are assumed to be ‘normal’ (such as typical traffic flow between two lanes) are extracted by using 27 objects and each object results in 3 types of trajectory vectors. Therefore, each of the 3 GM-HMMs is trained by using 27 trajectory vectors. Then, various ‘abnormal’ (such as reverse traffic flow or lane crossing) cases are generated and resulting trajectory vectors are tested via each of the GM-HMMs separately. The utilized test environment and trajectories are shown in Figure 6.6. The first model, GM_HMM_1, is trained by using first camera trajectories, whereas the second model, GM_HMM_2, is inputs the second camera trajectories. The last case, GM_HMM_1+2 is trained by using both trajectories. Each model has left-to-right connected 4 states, as shown in the Figure 6.8.

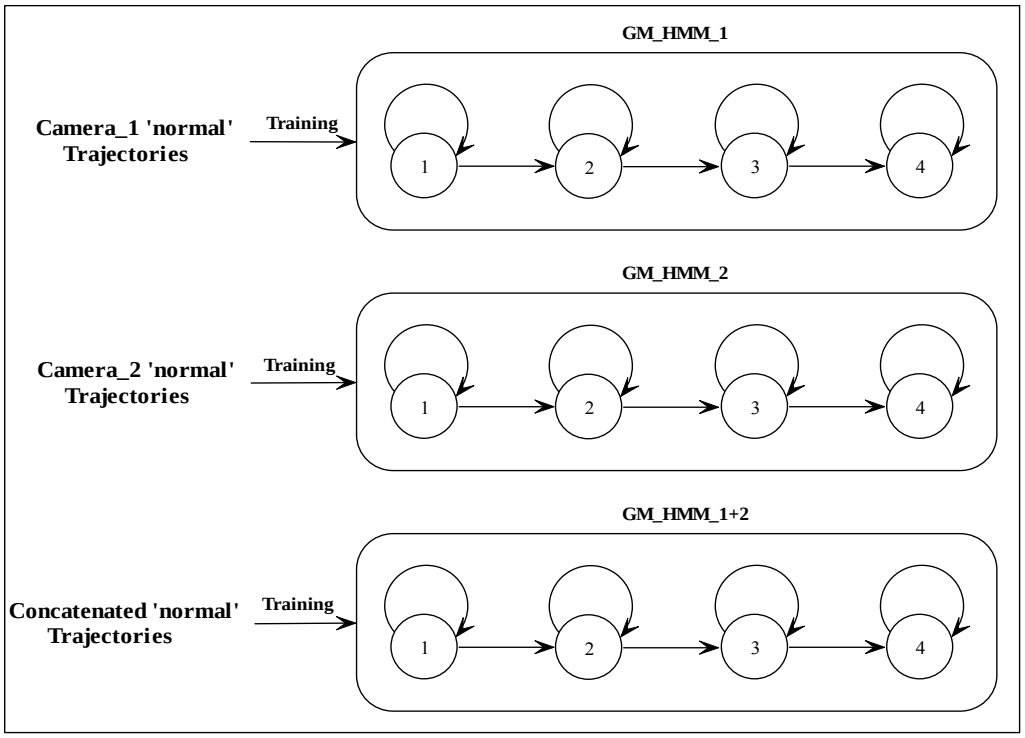


Figure 6.8 Trained Gaussian Mixture Hidden Markov Models

The resulting Viterbi distances [23] of the trajectories for the training object vectors are given in the Table 6.1. Average Viterbi distance of training objects for the three models is obtained as follows:

- Average distance to GM_HMM_1: 10.20
- Average distance to GM_HMM_2: 10.06
- Average distance to GM_HMM_1+2: 20.04

Viterbi distances of the most representative six trajectories, which are known to be 'abnormal', to the models are given in Table 6.2 and ratios of the test objects' Viterbi distances to the average Viterbi distances are given in the Table 6.3.

These ratios indicate that GM_HMM_1+2, which is trained by using concatenated trajectories, gives the best results to recognize 'abnormal' events for the given data set. In this data set, Objects 28 and 29 go in the opposite direction and Object 30 enters the FOV from sideways with respect to the training samples. Object 31 enters the FOV correctly but leaves slightly out of the 'normal' lane. Object 32 makes stop-and-go motion while Object 33 drops its speed during its motion. The first three trajectories, 28, 29, 30, are classified as 'abnormal' clearly, whereas 32nd and 33rd trajectories are classifies as 'normal' due to the time-warping property of the HMMs. Moreover, 31st one is also classified as 'normal', since its trajectory is not distinctive enough when compared to the training trajectories.

Table 6.1 Training case: Viterbi distances of trajectories for the training object vectors to the models

Object ID	Viterbi Distance to GM_HMM_1	Viterbi Distance to GM_HMM_2	Viterbi Distance to GM_HMM_1+2
1	10.0797	9.90183	19.7285
2	10.2908	10.1049	20.1867
3	10.2266	10.1233	20.1006
4	10.577	10.6716	21.2304
5	9.99018	9.84572	19.6763
6	10.0584	9.85901	19.6572
7	10.0608	9.88434	19.7496
8	10.2821	10.2472	20.3949
9	10.0773	9.8764	19.7181
10	10.3629	10.2508	20.3399
11	10.0322	9.86696	19.6382
12	10.0695	9.92222	19.7072
13	10.1321	9.95447	19.7818
14	10.2666	10.139	20.2119
15	10.2661	10.0629	20.0147
16	10.038	9.92932	19.6548
17	10.126	9.98202	19.7991
18	10.2134	10.108	19.9983
19	10.8046	10.5149	21.3008
20	10.4454	10.2919	20.333
21	10.111	9.90018	19.6983
22	10.1791	9.9294	19.9025
23	10.0511	10.0658	20.1564
24	10.1007	10.2248	20.248
25	10.3782	9.8986	19.9865
26	10.0308	9.9264	19.8682
27	10.0816	10.2139	20.1286

Table 6.2 Test case: Viterbi distances of test objects' trajectory vectors to the models

Object ID	Viterbi Distance to GM_HMM_1	Viterbi Distance to GM_HMM_2	Viterbi Distance to GM_HMM_1+2
28	20.4058	19.9481	45.1818
29	21.2409	19.7736	45.034
30	26.9917	24.7016	55.2278
31	10.7213	10.5773	21.2099
32	10.4648	10.5105	22.1852
33	10.1611	9.97222	19.7785

Table 6.3 Ratios of the Viterbi distances of test objects' trajectory vectors to the average Viterbi distances of training objects' trajectory vectors

Object ID	GM_HMM_1 ratios	GM_HMM_2 ratios	GM_HMM_1+2 ratios
28	2.00106	1.98236	2.25404
29	2.08295	1.96502	2.24666
30	2.64690	2.45474	2.75521
31	1.05137	1.05113	1.05812
32	1.02621	1.04448	1.10678
33	0.99643	0.99100	0.98671

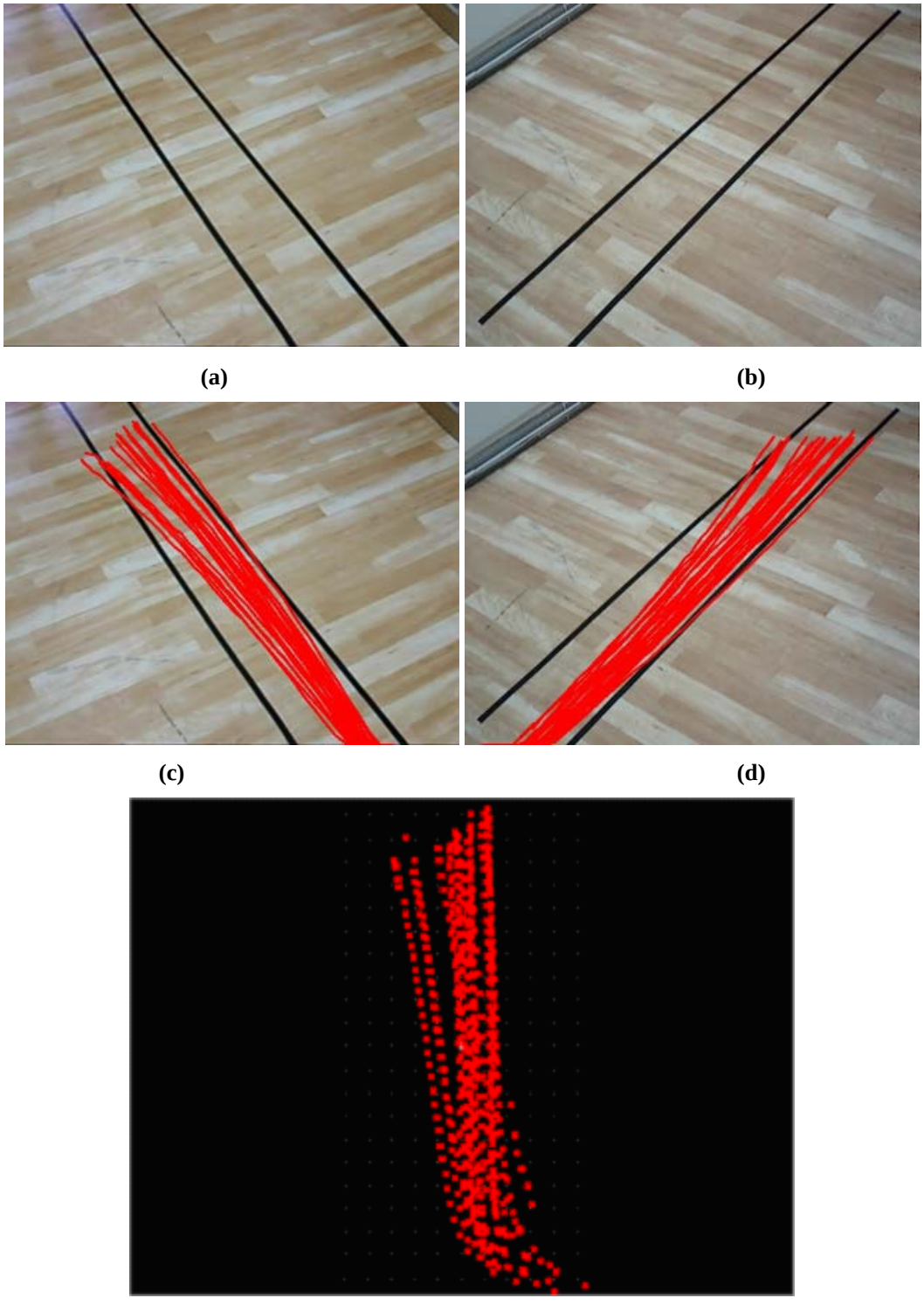


Figure 6.9 Test environment and training trajectories: (a) 1st camera view, (b) 2nd camera view and (c) 1st camera view trajectories of training samples (d) 2nd camera view trajectories of training samples (e) generated top-view trajectories of training samples

CHAPTER 7

CONCLUSION

7.1 Summary of the Thesis

In this thesis, novel methods for background modeling, tracking, occlusion handling and event recognition via multi-camera configurations are presented. Single camera employed versions of these methods are also examined. The proposed methods are mostly aimed for solving the problems in single camera surveillance systems by the help of multi-camera configurations.

In Chapter 2, fundamental building blocks of a single camera surveillance system are presented. First, four basic methods to segment moving objects are discussed and their performances in different conditions are compared. Some noise removal and shadow removal algorithms are also discussed. Second, four tracking algorithms are explained and their performances are compared. Finally, event recognition by using Gaussian Mixture Hidden Markov Models (GM-HMMs) is proposed for the mono-camera case. In this method, object locations in the image plane are used to construct trajectory vectors. Then, these vectors are used to train models. In the traffic data simulations, a different model is employed for each traffic lane, since each lane usually has different characteristics.

Moving object segmentation algorithms via multi-camera configurations are presented in Chapter 4. One of the key benefits of multi-camera usage is that it is possible to handle erroneous segmentation results by using auxiliary views. The first examined method is based on checking unanimity of the camera votes. Each camera has its own background model and votes for each pixel whether or not it is background. If one of the pixels is labeled as foreground in all off the views, then it is decided as foreground, otherwise it is forced to be background. In the second method, instead of unanimity, weighted voting is used. For each pixel, foreground decision is voted by each camera and final decision is given according to the sum of the all votes. Finally, background modeling by multivariate mixture of Gaussians is explained. In this method, a background model for each pixel is obtained by using multivariate Gaussians in which the each dimension is constructed by using the information coming from the different views.

In Chapter 5, a novel method for occlusion handling is proposed. One of the main problems of any surveillance system is occlusion, which might result in failure in all the aforementioned fundamental building blocks. The simplest way of handling occlusion is to generate an occlusion-free view. However, pixel-wise rendering methods are computationally demanding procedures for view generation and therefore, small (over-segmented) regions are used instead of pixels. In the proposed method, foreground masks (moving objects) in all views are over-segmented to obtain homogeneous regions by using RSST and K-means clustering methods. Then, segmented regions are matched between the views and region pairs are obtained. These region pairs are transferred to the occlusion-free view via a trifocal tensor by using the epipolar geometry. As a result, objects are converted into sparse set of regions in the generated view and they must be segmented to extract objects. Finally, graph-based clustering is performed to cluster similar regions which are belonging to distinct objects.

A multi-camera tracking and event recognition method is discussed in Chapter 6. Each object is tracked in different views by using a Kalman filter. As long as a

measurement can be found, separate trackers in different views continue tracking. When ‘no measurement’ case occurs in one of the views, the other view’s measurements are projected to ‘no measurement’ view via the homography between image planes and used to update corresponding tracker with a (projected) measurement. This method enables continuous tracking as long as one of the cameras can observe the object of interest. During tracking, trajectory vector of the object is extracted for all views. In proposed event recognition method, these trajectories are concatenated to generate a multi-view trajectory. These multi-view trajectories are used to train a GM-HMM. Then, incoming object trajectories are tested by this GM-HMM to classify their motion as ‘normal’ or ‘abnormal’, in order to detect incidents in the scene.

7.2 Discussions on Single Camera Surveillance

During moving object segmentation simulations, it is observed that one of the main problems of the single camera methods is the dynamic background regions. Frame differencing and the eigenbackground subtraction algorithms are found out to be sensitive to the dynamic backgrounds. Eigenbackground subtraction performs better, compared to the frame differencing but its update procedure demands a higher computational load. Parzen window based and Mixture of Gaussians based methods are more robust to these variations in the background. However, Parzen window based method has a quite demanding computational load. Also, computational cost of the MOG based moving object segmentation method increases as the number of Gaussians in the mixture increases, and 3-5 Gaussians found out to be convenient for each pixel’s model. Furthermore, time complexity of Parzen based and MOG based methods is strongly dependent to the size of the input image since pixel-wise modeling and comparison calculations are computational load demanding when compared to the frane differencing and eigenbackground subtraction.

The experimental results indicate that tracking via object association fails in the occluded scenes. Cam-shift and mean-shift trackers have promising results, when the occlusions are handled. However, they need correct initial models for the tracked objects and these algorithms would fail, if the objects enter the field of view, while occluding each other. Moreover, mean-shift tracker has a high computational complexity and observed as inappropriate for multi-object tracking. Also, cam-shift tracker fails when the target model has similar color values with background model due to its continiously adaptive nature which results in inconsistent variations in the search window of the tracked object. Kanade-Lucas-Tomasi Tracker is observed to be more robust to the occlusions and it could be the only solution for the cases in which the tracker is tested, especially during night surveillance. However, segmentation of the features for grouping them into individual objects is the most important drawback of the KLT method. It is observed that features which belong to the different objects, while close to each other, cannot be separated, when they move with similar velocities.

Single camera event recognition results imply that single HMM fails to model the entire scene, if there are more than one ‘normal’ event types for same measurement. In the traffic scenario, each lane has different trajectory vectors that can be named as ‘normal’. Therefore, an event that can be classified as ‘normal’ for one of the lanes might be ‘abnormal’ for another lane, and hence, more than one HMM is needed to model traffic. Experimental results from different scenarios in typical traffic videos indicate that the classifiers are successful to recognize ‘abnormal’ events.

7.3 Discussions on Multi-Camera Surveillance

In the proposed multi-camera moving object segmentation methods, the views are related via homography to each other and objects are assumed to be planar on the ground plane. Therefore, heights of the objects must be negligible compared to

the camera altitudes to satisfy this assumption and cameras must be mounted on devices at high altitudes. Otherwise, as shown during the simulations, the projection errors due to the plane assumption results in slight distortions in the foreground masks. Apart from this fact, the objects, which are occluded by static background obstacles in one of the views, can be segmented as foreground via the proposed methods, if other views can see the object. Also, false segmented regions in one of the views can be eliminated if the same region is correctly segmented in the other views.

During the simulations of the occlusion handling method, it is observed that the proposed method gives succesfull results when objects are partially occluded or have different colors with respect to each other. Similar colored objects could also be separated, if one of the cameras is able to see the objects without occlusion. However, for the cases, where similar colored objects occlude each other heavily or single object has different colors on it, system may under-segments or over-segments the objects. Moreover, performances of the segmentation algorithms, RSST and K-means, are found out to be similar, when final results are considered. However, K-means segments the data much faster than RSST and is observed to be better suited for any real-time application.

Multi-camera tracking simulation results imply that once the same objects in different views are matched, the examined method gives promising results. Therefore, clear masks of foreground objects must be segmented, when they initially appear in the common field of view. The obvious advantage of the proposed system over single camera trackers is due to continues tracking as long as one of the cameras is able to view the object. As shown in the simulation results, Kalman tracker (without assistance) failed to track the object passing behind an obstacle and new Kalman trackers are mistakenly initiated since lost objects are incorrectly identified as appearing for the first time. Moreover, it is observed that GM-HMM, which is trained by the trajectories, extracted via multi-

cameras, yields better results, when the classification of ‘abnormal’ events is considered.

7.4 Future Work

The proposed multi-camera methods are designed for the static camera configurations and must be calibrated after installation. This calibration is performed by an operator via 3D landmarks and must be automated. Moreover, many surveillance systems use active PTZ-cameras (pan-tilt-zoom) that can be controlled remotely, and these systems have advantages when static cameras are considered. In the future work, self calibration of these cameras should be possible to be performed automatically.

Occlusion handling and tracking are considered as separate modules. An automated combination unit should be incorporated into a surveillance system, which will automatically recognize the occlusions and performs occlusion handling while tracker continues to track resulting separated objects.

REFERENCES

[1] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4: Real-Time Surveillance of People and Their Activities", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 809-830, 2000.

[2] A. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric Model for Background Subtraction", *Proceedings of IEEE International Conferance on Computer Vision, Frame-Rate Workshop*, 1999.

[3] W. E. L. Grimson and C. Stauffer, "Adaptive Background Mixture Models for Real-time Tracking", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 22-29, 1999.

[4] P. KaewTraKulPong and R. Bowden, '"An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection", *Proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[5] N. Oliver, B. Rosario, and A. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions", *International Conference on Vision Systems*, 1999.

[6] M. Piccardi, "Background Subtraction Techniques: a Review", *Proceedings of IEEE SMC International Conference on Systems, Man and Cybernetics*, Vol. 4, pp. 3099-3104, 2004.

[7] B. B. Orten, *Moving Object Identification and Event Recognition in Video Surveillance Systems*, M.S. Thesis, Middle East Technical University, Turkey, July 2005.

[8] R. Jain, R. Kasturi, and B. G. Schunk, *Machine Vision*, McGraw-Hill, International Editions, 1995.

[9] C. Jiang and M. O. Ward, “Shadow Identification", *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 606-612, 1992.

[10] F. M. Porikli and J. Thornton, “Shadow Flow: A Recursive Method to Learn Moving Cast Shadows”, *IEEE International Conference on Computer Vision*, Vol. 1, pp. 891-898, October 2005.

[11] I. Mikic, P. Cosman, G. Kogut, and M. Trivedi, “Moving Shadow and Object Detection in Traffic Scenes”, *International Conference on Pattern Recognition*, Vol. 1, pp. 321-324, 2000.

[12] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, “Detecting Objects, Shadows and Ghosts in Video Streams by Exploiting Color and Motion Information”, *Proceedings of IEEE Conference on Image. Analysis and Processing*, 2001.

[13] D. Comaniciu, V. Ramesh, and P. Meer, "Real-Time Tracking of Non-Rigid Objects Using Mean Shift", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 142-149, June 2000.

[14] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-Based Object Tracking”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 5, pp. 564-575, May 2003.

[15] G. R. Bradski, “Real Time Face and Object Tracking as a Component of a Perceptual User Interface”, *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision*, pp. 214, October 1998.

[16] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features", *Carnegie Mellon University Technical Report CMU-CS-91-132*, April 1991.

[17] J. Y. Bouguet, “Pyramidal Implementation of the Lucas-Kanade Feature Tracker”, *Microsoft Research Labs Tech. Report*, 1999.

[18] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A Real-Time Computer Vision System for Measuring Traffic Parameters", *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 495-501, 1997.

[19] T. Jehan, *Creating Music by Listening*, PhD Thesis, Massachusetts Institude of Technology, September 2005.

[20] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems", *Transactions on the ASME Journal of Basic Engineering*, Vol. 82, pp. 35-45, 1960.

[21] P. S. Maybeck,. *Stochastic Models, Estimation, and Control: Volume 1*, Academic Press, New York, 1979.

[22] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Second Edition, Wiley, 2001.

[23] L. Rabiner.and B. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[24] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, UK, 2003.

[25] E. Tola, *Multiview 3D Reconstruction of a Scene Containing Independently Moving Objects*, M.S. Thesis, Middle East Technical University, Turkey, August 2005.

[26] D. Eppstein, "Spanning Trees and Spanners", *Chapter 9 in Handbook of Computational Geometry (Ed. J. R. Sack and J. Urrutia)*, pp. 425-461, 2000.

[27] B. Y. Wu and K. M. Chao, *Spanning Trees and Optimization Problems*, CRC Press, 2004.

[28] J. Black, *Multi-view Image Surveillance and Tracking*, PhD. Thesis, City University School of Engineering, London, April 2004.

[29] M. Harville, G. Gordon, and J. Woodfill, “Adaptive Video Background Modeling Using Color and Depth”, *International Conference on Image Processing*, Oct. 2001.

[30] B. Goldlucke and M. A. Magnor, “Joint 3D Reconstruction and Background Separation in Multiple-views Using Graph Cuts”, *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 683, June 2003.

[31] V. Kolmogorov and R. Zabih, “Multi-camera Scene Reconstruction via Graph Cuts”, *7th European Conference on Computer Vision*, pp. 82-96, 2002.

[32] C. Eveland, K. Konolige, and R. C. Bolles, “Background Modeling for Segmentation of Video-rate Stereo Sequences”, *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 266-271, June 1998.

[33] Y. A. Ivanov, A. F. Bobick, and J. Liu, “Fast Lighting Independent Background Subtraction”, *International Journal of Computer Vision*, Vol. 37, No. 2, pp. 199-207, 2000.

[34] S. N. Lim, A. Mittal, L. S. Davis, and N. Paragios, “Fast Illumination-Invariant Background Subtraction using Two-Views: Error Analysis, Sensor Placement and Applications”, *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.

[35] A. M. Tekalp and S. L. Dockstader, "Tracking Multiple Objects in the Presence of Articulated and Occluded Motion", *Proceedings of IEEE Workshop on Human Motion*, pp. 88-95, December 2000.

[36] C. C. C Pang, W. W. L. Lam, and N. H. C.Yung, “A Novel Method for Resolving Vehicle Occlusion In A Mononular Traffic Image Sequence”, *IEEE Transactions on Intelligent Transportation Sysytems*, Vol. 5, No. 3, pp. 129-141, September 2004.

[37] X. Song and R. Nevatia, “A Model-based Vehicle Segmentation Method for Tracking”, *Proceedings of the 10th IEEE International Conference on Computer Vision*, Vol. 2, pp. 1124-1131, 2005.

[38] N. K. Kanhere, S. J. Pundlik, and S. Birchfield, "Vehicle Segmentation and Tracking from a Low-Angle Off-Axis Camera", *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 1152-1157, 2005.

[39] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer, "Multi-camera Multi-person Tracking for EasyLiving", *3rd IEEE International Workshop on Visual Surveillance*, 2000.

[40] K. Kim and L. S. Davis, "Multi-Camera Tracking and Segmentation of Occluded People on Ground Plane using Search-Guided Particle Filtering", *European Conference on Computer Vision*, 2006.

[41] A. Mittal and L. S. Davis, "M2Tracker: A Multi-view Approach to Segmenting and Tracking People in a Cluttered Scene", *International Journal of Computer Vision*, Vol. 51, No. 3, 2003.

[42] A. Mittal and L. Davis, "Unified Multi-camera Detection and Tracking using Region-matching", *Proceedings of IEEE Workshop on Multi-Object Tracking*, pp. 3-10, July 2001.

[43] J. Orwell, P. Remagnino, and G. A. Jones, "Multi-Camera Color Tracking", *Proceedings of the 2nd IEEE Workshop on Visual Surveillance*, 1998.

[44] O. Ersoy, *Image Segmentation with Improved Region Modeling*, M.S. Thesis, Middle East Technical University, Turkey, December 2004.

[45] J. F. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 679-698, 1986.

[46] C. Harris and M. A. Stephens, “Combined Corner and Edge Detector”, *4th Alvey Vision Conference*, pp. 147-151, 1988.

[47] S. M. Smith and J. M. Brady, “SUSAN-A New Approach to Low Level Image Processing”, *International Journal of Computer Vision*, pp. 45-78, 1997.

[48] K. Mikolajczyk and C. Schmid, “An Affine Invariant Interest Point Detector”, *Proceedings of the 7th European Conference on Computer Vision*, Vol. 1, pp. 128-142, May 2002.

[49] C. Schmid, R. Mohr, and C. Bauckhage, “Comparing and Evaluating Interest Points”, *IEEE International Conference on Computer Vision*, pp. 230-235, 1998.

[50] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91-110, 2004.

[51] H. Moravec, “Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover”, *Carnegie-Mellon University Robotics Institute Technical Report CMU-RI-TR-3*, 1980.

[52] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson, "Advances in Cooperative Multi-Sensor Video Surveillance", *DARPA Image Understanding Workshop*, pp. 3-24, November 1998.

[53] W. Hu, T. Tan, L. Wang, and S. Maybank, "A Survey on Visual Surveillance of Object Motion and Behaviors", *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, Vol. 34, No. 3, pp. 334-352, 2004.

[54] V. Kettnaker and R. Zabih, “Bayesian Multi-camera Surveillance”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 253-259, 1999.

[55] S. L. Dockstader and A. M. Tekalp, “Multiple-camera Tracking of Interacting and Occluded Human Motion”, *Proceeings of IEEE*, Vol. 89, No. 10, pp. 1441-1455, Oct. 2001.

[56] R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade, “Algorithms for Cooperative Multi-sensor Surveillance”, *Proceedings of IEEE*, Vol. 89, pp. 1456-1477, Oct. 2001.

[57] J. Black, T. Ellis, and P. Rosin, “Multi-view Image Surveillance and Tracking”, *Proceedings of the Workshop on Motion and Video Computing*, 2002.

[58] J. Black and T. Ellis, "Multi-camera Image Tracking", *Image and Vision Computing, Elsevier*, 2005.

[59] T. H. Chang and S. S. Gong, “Tracking Multiple People with a Multi-camera System”, *IEEE Workshop on Multi-Object Tracking*, pp. 19-28, July 2001.

[60] T. H. Chang, S. S. Gong, and E. J. Ong, “Tracking Multiple People Under Occlusion Using Multiple Cameras”, *British Machine Vision Conference*, September 2000.

[61] G. T. Kogut and M. Trivedi, “Maintaining the Identity of Multiple Vehicles as They Travel Through a Video Network”, *IEEE Workshop on Multi-Object Tracking*, pp. 29-34, July 2001.

[62] O. Javed, S. Khan, Z. Rasheed, and M. Shah, “Camera Handoff: Tracking in Multiple Uncalibrated Stationary Cameras”, *IEEE Workshop on Human Motion*, pp. 113-120, December 2000.

[63] O. Javed, Z. Rasheed, A. Alatas, and M. Shah “KNIGHT: A Real Time Surveillance System for Multiple Overlapping and Non-Overlapping Cameras”, *International Conference on Multimedia and Expo*, 2003.

[64] O. Javed, Z. Rasheed, K. Shafique, and M. Shah “Tracking Across Multiple Cameras With Disjoint Views”, *IEEE International Conference on Computer Vision*, pp. 952-957, 2003.

[65] S. Khan, O. Javed, Z. Rasheed, and M. Shah, “Human Tracking in Multiple Cameras”, *IEEE International Conference on Computer Vision*, pp. 331-337, July 2001.

[66] S. Khan, O. Javed, and M. Shah, “Tracking in Uncalibrated Cameras with Overlapping Field of View”, *2nd International Workshop on Performance Evaluation of Tracking and Surveillance*, December 2001.

[67] R. Polana and R. Nelson, “Low Level Recognition of Human Motion”, *Proceedings of IEEE Workshop Motion of Non-Rigid and Articulated Objects*, pp. 77-82, 1994.

[68] C. Stauffer and W. E. L. Grimson, "Learning Patterns of Activity Using Real-Time Tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 747-757, August 2000.

[69] K. K. Lee, M. Yu, and Y. Xu, “Modeling of Human Walking Trajectories for Surveillance”, *Intelligent Robots and Systems*, Vol. 2, pp.1554-1559, Oct. 2003.

[70] S. Rao and P. S. Sastry, "Abnormal Activity Detection in Video Sequences using Learnt Probability Densities", *Conference on Convergent Technologies for Asia-Pacific Region*, Vol. 1, pp. 369-372, Oct. 2003.

[71] M. Brand, N. Oliver, and A. Pentland, “Coupled HMMs for Complex Action Recognition”, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 994-999, 1997.

[72] F. Porikli and X. Li, “Traffic Congestion Estimation Using HMM Models Without Vehicle Tracking”, *IEEE Intelligent Vehicle Symposium*, pp. 188-193, 2004.

[73] F. Bashir, A. Khokhar, and D. Schonfeld, "Automatic Object Trajectory-Based Motion Recognition using Gaussian Mixture Models", *IEEE International Conference on Multimedia and Expo*, July 2005.

[74] F. Bashir, A. Khokhar, and D. Schonfeld, "HMM-Based Motion Recognition System using Segmented PCA", *IEEE International Conference on Image Processing*, Sept 2005.

[75] S. Kullback and R. A. Leibler, "On Information and Sufficiency", *Annals of Mathematical Statistics*, Vol. 22, No. 1, pp. 79-86, March 1951.

[76] 2nd IEEE International Workshop on Performace Evaluation of Tracking and Surveillance, http://www.cvg.cs.rdg.ac.uk/PETS2001/pets2001-dataset.html, (08.11.2006).