MULTIPLE HYPOTHESIS TRACKING FOR MULTIPLE VISUAL TARGETS

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

BY

BURCU TÜRKER

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

APRIL 2010

Approval of the Thesis

"MULTIPLE HYPOTHESIS TRACKING FOR MULTIPLE VISUAL TARGETS"

Submitted by **BURCU TÜRKER** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen

Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İsmet Erkmen

Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. M. Kemal Leblebicioğlu

Supervisor, **Electrical and Electronics Engineering, METU** _____

Prof. Dr. Gözde Bozdağı Akar

Co-supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members

Prof. Dr. A. Aydan Erkmen
Electrical and Electronics Engineering, METU _____

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

Prof. Dr. Uğur Halıcı
Electrical and Electronics Engineering, METU _____

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

Umur Akıncı (M.Sc.)
MGEO, ASELSAN A.Ş. _____

Date: 28.04.2010

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 : Burcu Türker

Signature :

ABSTRACT

MULTIPLE HYPOTHESIS TRACKING FOR MULTIPLE VISUAL TARGETS

Türker, Burcu

M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. M. Kemal Leblebicioğlu
Co-supervisor: Prof. Dr. Gözde Bozdağı Akar

April 2010, 119 pages

Visual target tracking problem consists of two topics: Obtaining targets from camera measurements and target tracking. Even though it has been studied for more than 30 years, there are still some problems not completely solved. Especially in the case of multiple targets, association of measurements to targets, creation of new targets and deletion of old ones are among those. What is more, it is very important to deal with the occlusion and crossing targets problems suitably. We believe that a slightly modified version of multiple hypothesis tracking can successfully deal with most of the aforementioned problems with sufficient success. Distance, track size, track color, gate size and track history are used as parameters to evaluate the hypotheses generated for measurement to track association problem whereas size and color are used as parameters for occlusion problem. The overall tracker has been fine tuned over some scenarios and it has been observed that it performs well over the testing scenarios as well. Furthermore the performance of the tracker is

analyzed according to those parameters in both association and occlusion handling situations.

Keywords: Visual Surveillance, Mixture of Gaussians Method, Moving Object Detection, Multiple Hypothesis Tracking.

ÖZ

ÇOKLU HİPOTEZ YÖNTEMİ İLE ÇOKLU GÖRSEL HEDEF TAKİBİ

Türker, Burcu

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. M. Kemal Leblebicioğlu
Ortak Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Nisan 2010, 119 sayfa

Görsel hedef takibi problemi kamera ölçümlerinden hedeflerin elde edilmesi ve hedef takibi olmak üzere iki temel konudan oluşur. 30 yılı aşkın süredir bu konuda çalışmalar yapılmaktadır ancak halen bazı problemler tamamen çözülmemiştir; bunların başında birden fazla hedef takibi, ölçümlerin hedefler ile eşleştirilmesi, yeni hedeflerin oluşturulması ve eski hedeflerin silinmesi gibi problemler yer almaktadır. Ayrıca hedeflerin örtüşmesi ve çapraz geçişleri de hedef takibinde çözümlenmesi gereken önemli konulardandır. Çoklu hipotez yönteminin bir miktar uyarlanması ile sözü geçen problemlerin çoğunun çözümünde önemli ölçüde başarı elde edileceğine inanmaktayız. Ölçüm-hedef eşleştirme hipotezlerini değerlendirmek için uzaklık, hedef büyüklüğü, hedef rengi, kapı genişliği ve hedef geçmişi parametre olarak kullanılırken hedeflerin örtüşmesi problemini çözmek için hedef rengi ve hedef büyüklüğü değerlendirme parametresi olarak kullanılmıştır. Tasarlanan hedef takipçisi bazı senaryolar üzerinde ayarlandıktan sonra test senaryoları üzerinde denenmiş ve iyi bir performans gösterdiği gözlenmiştir. Ayrıca

hedef takipçisinin ölçüm-hedef eşleştirme ve hedeflerin örtüşmesi problemlerinin çözümündeki başarısı bu parametrelere göre analiz edilmiştir.

Anahtar Kelimeler: Görsel Gözetim, Gauss'ların Karışımı Yöntemi, Hareketli Nesnelerin Tespiti, Çoklu Hipotez Yöntemi.

To My Family and Özden

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to my supervisor Prof. Dr. M. Kemal Leblebicioğlu and Prof. Dr. Gözde Bozdağı Akar for their supervision, guidance and encouragement throughout this study.

I would like to also express my thanks to my friends, my colleagues and Aselsan A.Ş. for their precious support and fellowship.

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

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Visual surveillance is one of the popular research areas in computer vision as a result of the society's increasing needs especially in security issues and the developments in computer technology. Visual surveillance systems are considered as a solution to security problems in both civilian and military areas. Crowded places such as airports, train stations, banks and strategically critical regions like country borders are the major application areas of these systems. Furthermore, in traffic monitoring, visual surveillance systems are becoming more important because of their additional ability of recording the violations of traffic rules. As computer technology improves, processor speeds and capacities increase and this makes implementing complex and computationally loaded algorithms possible. Furthermore, as camera technology develops, output quality and zoom ability of cameras increase and their sizes gets even smaller. Therefore cameras can be mounted anywhere and the detection of human's face mimics is even possible.

Visual surveillance systems monitor and report the activities in a specified area by analyzing the data coming from the video cameras or visual sensors which are placed in the region of interest. These cameras can be fixed or placed on pan-tilt devices. The data gathered from video cameras is analyzed by computers or human operators or both. In traditional surveillance systems, human effort is utilized for the analysis of the activities from camera outputs, whereas, computers are used for this purpose in automated systems. In recent years, automated surveillance systems are taking the place of traditional ones because of their advantages such as being cheaper and more reliable. A computer can analyze the data from many cameras at

the same time, however many human operators are needed to do the same job; therefore from this respect, traditional systems are more expensive. Besides, in parallel with the technological developments, there is a decrease in camera and computer costs. Furthermore, the performance of traditional systems is highly dependent on human operators' concentration, making these systems less reliable.

An automated visual surveillance system consists of three basic steps which are information coming from one video camera or from well-spaced camera nets, motion detection and target tracking. In some applications, data (information coming from many cameras) fusion step is performed in the first place whereas in some other applications it is performed in late steps. Furthermore, in some surveillance implementations, motion detection and target tracking steps are realized together as one step. There may be some additional abilities of the system such as motion or target recognition and action decision (specified motion or target recording and alarm giving on specified conditions). The general flow of visual surveillance systems is shown in Figure 1-1.

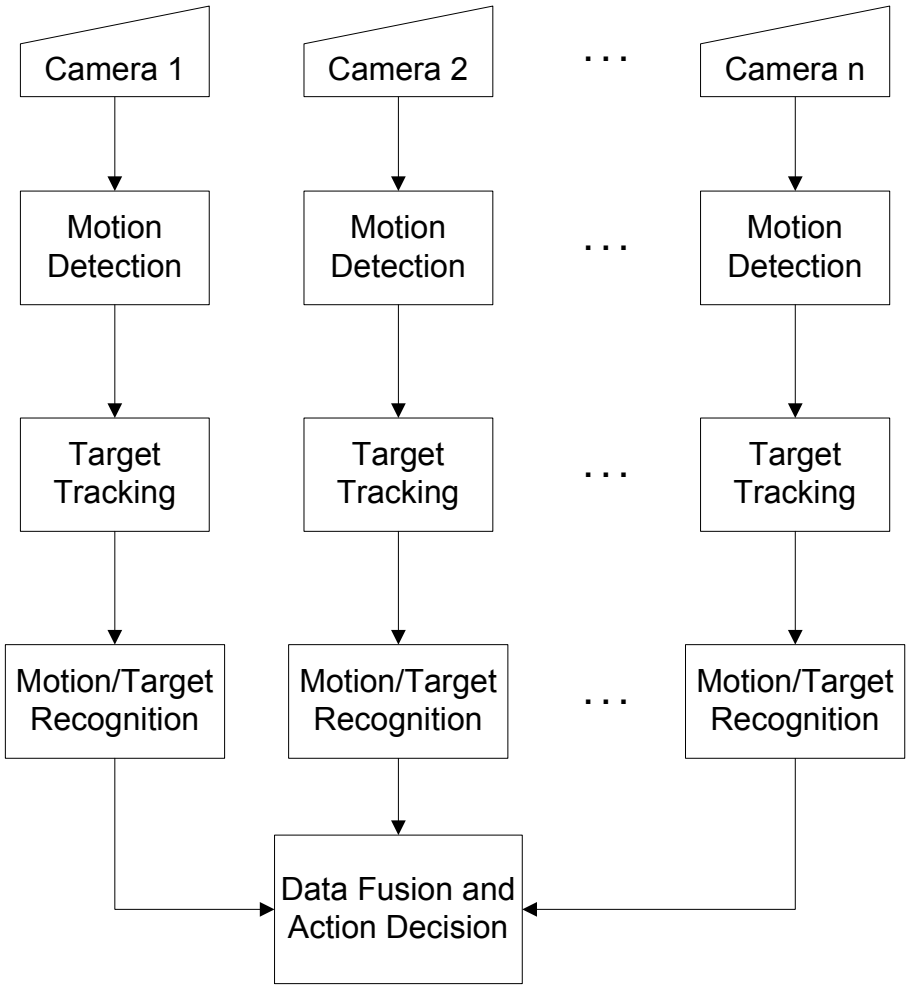


Figure 1-1 General flow of visual surveillance systems

The researches on surveillance systems are mostly concentrated on improving the performance of the system in terms of motion detection by making it robust to the dynamic changes in the environment and target tracking by solving the problems such as occlusions and tracking targets passing behind an object.

1.1 Scope of the Thesis

The aim of the thesis is to develop an automatic multiple object detection and tracking system. The developed system is composed of two parts: 1) Moving object detection based on Gaussian Mixture Model (GMM) and 2) Moving object tracking based on Multiple Hypothesis Tracking (MHT).

In moving object detection, first the foreground objects are detected by GMM. The algorithm is tested with different learning rates. Then a statistical method is applied for shadow removal and after that erosion, dilation and connected component labeling operations are performed on these detected foreground parts.

In moving object tracking part of the system, multiple hypothesis tracking method is applied which is capable of tracking multiple objects with track initiation, track confirmation and track deletion. Hypothesis based MHT is implemented to solve the observation-to-track association problem of multiple target tracking. In addition to distance feature as in standard MHT [24], color, size, gate size and track history features are also used for solving the association problem. Problematic cases including occlusion, tracking targets passing behind an object and tracking temporarily undetected targets are handled.

The performance of the developed system is analyzed through extensive simulations. First analysis aims to examine the effects of the duration of the background training period on tracking performance. The second analysis aims to see the sensitivity of the tracking performance on the features that are utilized for observation-to-track association. Finally, the third analysis examines the sensitivity of the occlusion handling performance on the features that are utilized for the separation of occlusion participants.

1.2 Outline of the Thesis

In Chapter 2, the related studies on the moving object detection and moving object tracking parts of visual surveillance systems are presented.

In Chapter 3, moving object detection using mixture of Gaussians method is described and the results of the implementation of the proposed method are presented. Shadow removal using a statistical method, noise removal using morphological operations such as erosion and dilation and connected component labeling are also described in this chapter.

In Chapter 4, the steps of moving object tracking using multiple hypothesis tracking method is presented in detail and the results of the implementation of the proposed method are presented. Track confirmation and occlusion handling issues are also described in this chapter.

In Chapter 5, the details of the performance analysis of the developed system are presented. In the first part of the analysis, the effects of the duration of the background training period on tracking performance; in the second part, the sensitivity of the tracking performance on the features that are utilized for observation-to-track association and in the third part, the sensitivity of the occlusion handling performance on the features that are utilized for separation score calculation are examined.

In Chapter 6, the summary of the thesis is presented and future work is discussed.

CHAPTER 2

A SHORT SURVEY ON VISUAL SURVEILLANCE

Visual surveillance systems consist of three basic parts which are moving object detection, moving object tracking and multiple target tracking. This chapter summarizes the latest studies in the literature related with the scope of the thesis.

2.1 Moving Object Detection

The first part of a visual surveillance system is generally moving object detection which is the process of distinguishing the moving parts of video streams. The following parts of the system use the output of this part; therefore the performance of the whole system is highly dependent on the performance of the moving object detection part. There may be some difficulties in detection of moving objects such as repetitive motion and unstable illumination depending on the input video stream. The methods of moving object detection are grouped under three main categories according to [1], [7]: temporal differencing, background subtraction and optical flow.

2.1.1 Temporal Differencing

Temporal differencing method uses the intensity differences of each pixel between consecutive frames to make a decision whether a pixel is part of background or a moving object. The pixels that have differences greater than a threshold are considered as foreground where as the pixels with smaller differences are considered as background. Here, the determination of the threshold value has great

importance. The simplest way to determine the moving parts of a video frame using this method is to use the intensity difference of consecutive frames pixel by pixel [12] and motion image detection is formulated in (1-1) and (1-2).

$$\Delta_n = \left| I_n - I_{n-1} \right| \tag{1-1}$$

$$M_n = \begin{cases} I_n, & \Delta_n \geq T \\ 0, & \Delta_n < T \end{cases} \tag{1-2}$$

where I_n be the intensity of the n^{th} frame, Δ_n be the intensity difference of two consecutive frames, M_n be the motion image and T be the threshold value.

Here the determination of the motion image detection threshold value is important which is dependent on the application and can be set experimentally. According to [7] the disadvantage of this technique is its dependency on the speed of the motion image whereas it has the advantage of being adaptive to dynamic environments and computationally efficient.

2.1.2 Background Subtraction

Background subtraction uses a background model to detect moving parts of the video frames and is one of the most common moving object detection methods. This method generates a background model that designates the video sequence without moving parts and performs the comparison of the current frame with the background model. Background subtraction approaches are categorized according to their way of background modeling.

Pfinder [21] which is a real-time person finder system models each pixel of the background using Gaussian distribution. The color of the pixels is modeled in terms of their mean and covariance values. A statistical color and shape model is used for the representation of the moving parts. Pfinder also uses an adaptive filter to compensate for illumination changes and the differences in the background model caused by the motion of the human.

Instead of modeling each background pixel with a single Gaussian model, [22] proposes a method, mixture of Gaussians, that models every background pixel with a number of Gaussian distributions. Based on the persistence and the variance of each of the Gaussians of the mixture, they determine which Gaussians may correspond to background colors. This method performs well in backgrounds with cluttered regions and slowly moving parts. The details of the mixture of Gaussians method is given in 3.1.1.

Yang et al. [5] proposes a method using two-layer Gaussian mixture model for the case of changing environment. One of the layers models the pixels that change gradually whereas the other layer models the pixels that change abruptly. This method models the dynamic scene accurately and is good at transparent problem solution.

Haritaoğlu [26] models each pixel of the background with its minimum and maximum intensity values and maximum intensity difference between consecutive frames.

In this study mixture of Gaussians method is chosen as the moving object detection method.

2.1.3 Optical Flow

Optical flow based motion detection [26] uses characteristics of flow vectors of moving objects over time to detect moving regions in an image sequence. For example, Meyer et al. [34] compute the displacement vector field for the extraction of articulated objects. This kind of methods are computationally complex and noise sensitive, however they have the advantage of being able to detect the moving objects even in the case of moving cameras.

2.2 Moving Object Tracking

The second part of visual surveillance systems is moving object tracking which follows the detection of moving objects in video frames. The main idea of this step

is to obtain the association of moving objects between consecutive frames. These association approaches may differ from each other depending on the methods they utilize to correlate the moving objects. According to [26], tracking methods are divided into four major categories: region-based tracking, active-contour-based tracking, feature-based tracking and model-based tracking.

2.2.1 Region-Based Tracking

In region-based tracking methods moving object region differences between the consecutive frames are considered.

The study by McKenna on tracking groups of people [13] is a good example of region-based tracking in which tracking is realized at three levels of abstraction: regions, people and groups. McKenna et al. [13] defines regions as connected components that have been tracked for at least a number frames with a bounding box, a support map, a timestamp and a tracking status. People are defined as one or more regions grouped together with an appearance model based on color; and groups are defined as one or more people grouped together. Track initializations, deletion, merging and splitting issues are performed based on regions. It is also stated that region-based tracking together with color appearance model gives successful results even in occlusion cases.

2.2.2 Active-Contour-Based Tracking

Active contour-based tracking algorithms track objects by representing their outlines as bounding contours and updating these contours dynamically in successive frames [26].

Tracking using “snakes” is one of the active-contour-based tracking methods and it is based on snake energy minimization. Snake active-contour models, firstly used by Kass et al. [15] as a basis for interactively matching 3D models to images and tracking.

Paragio and Deriche [14] propose a framework that links the minimization of a geodesic active contour objective function to the detection and the tracking of moving objects and this framework is implemented using the level set method and can successfully deal with the challenging problem of tracking non-rigid objects that cannot be easily parameterized.

2.2.3 Feature-Based Tracking

Feature-based tracking algorithms are mostly used in motion recognition studies. This kind of algorithms are defined [26] as the algorithms that perform recognition and tracking of objects by extracting elements, clustering them into higher level features and matching the features between images. The features that are used for correspondence are color, perimeter, area, centroid, distance.

A robust method for the recognition of activities like walking is proposed in [28] where a feature-based tracking algorithm is used. Tracking of moving objects is performed following the centroid features of the objects between consecutive frames and the recognition part of the method is based on low-level features of motion.

2.2.4 Model-Based Tracking

A tracking algorithm that combines motion and appearance information into an observation model and uses a particle filter framework for tracking the objects in subsequent frames is proposed in [1]. A simple adaptive Markov model is used as the adaptive state transition model with adaptive velocity and noise. Next, the estimation of the unknown state vector is performed using a particle filter. It is stated that this algorithm is very robust even in challenging tracking conditions like static occlusion and cluttering background.

2.3 Multiple Target Tracking

Almost all of the visual surveillance systems deals with multiple targets and multiple target tracking part is one of the most important parts of these systems. The

detection of moving objects (observations) is followed by association of observations with targets. The accuracy of this association has great importance since next states of the associated targets are predicted using the output of this correspondence.

Multiple target tracking methods evaluates the observations in each frame and decides whether the observation initiates a new target or updates an existing target. While making this decision the observations within the predicted area of the targets are taken into consideration.

The popular and basic target association problem is closely spaced multiple targets case in which target 1 and target 2 shares two observations in their predicted areas [25]. This problematic case is shown in Figure 2-1, where P1 and P2 denote the predicted positions of the targets and O1, O2 and O3 denote the positions of the observations.

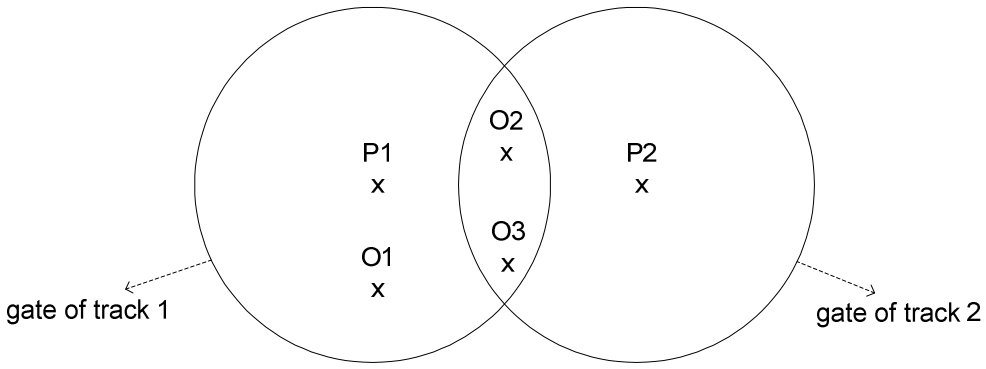


Figure 2-1 Closedly spaced multiple targets

A conventional MTT system consists of observation-to-track association, track maintenance, filtering and prediction and gating computations stages [18]. The block diagram of this system is shown in Figure 2-2.

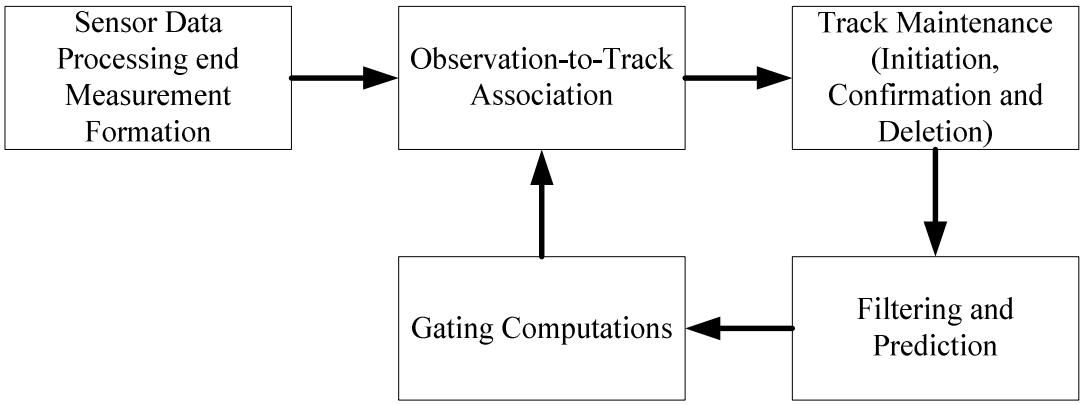


Figure 2-2 Basic elements of a conventional MTT system

There are many solutions of observation-to-track association problem; three of which are explained here: Nearest Neighbor Method, Global Nearest Neighbor Method and Multiple Hypothesis Tracking Method. The performance of these methods can be evaluated according to their simplicity, computational load and the concerned tracking problem itself.

2.3.1 Global Nearest Neighbor

Global Nearest Neighbor (GNN) method [18] finds the most likely assignment of input observations to existing tracks. The term global is used to refer to the fact that the assignment is made considering all possible (within gates) associations under the constraint that an observation can be associated with at most one track. Blackman [18] defines gate as the predicted area in which the track can appear based on the maximum acceptable measurement plus tracking prediction error magnitudes.

For the case in Figure 2-1, O1 would be assigned to track 1, O2 would be assigned to track 2 and O3 would initialize a new track called track 3. Here, the most important assumption is an observation can only be produced by a single target. Blackman describes the targets having no common observation as compatible targets. According to [18] Global Nearest Neighbor assignment solution contains only compatible tracks and unassigned observations initiate new track candidates. A

track candidate is validated after a number of consecutive frames in which the existence of that track is confirmed. The number of the frames needed to confirm its existence can be set to 3. In a similar manner, an existing track can be deleted after confirming its absence with a number, say 5, of consecutive frames. Some other rules can be introduced for track confirmation and deletion. After assignment of observations to tracks and track confirmation is performed, next positions of the tracks are estimated. Kalman Filter is a widely used technique for next state estimation.

2.3.2 Nearest Neighbor

Nearest Neighbor (NN) method [18] finds the closest observation to existing tracks. Different from GNN, an observation can be shared with another track, in other words, tracks do not have to be compatible with each other. The observation-to-track assignment is performed as in GNN concerning the observations in the gate of the tracks.

For the example in Figure 2-2, the closest observations are chosen for each track. Let us assume the nearest observation to track 1 is O1 and track 2 to O2; then, O1 is assigned to track 1, O2 is assigned to track 2 and O3 would initialize a new track called track 3. The remaining part of the method is the same as GNN Method. A track candidate is validated after a number of consecutive frames at which the existence of that track is confirmed. The number of the frames needed to confirm its existence can be set to 3. In a similar manner, an existing track can be deleted after confirming its absence with a number of consecutive frames, say 5. Some other rules can be introduced for track confirmation and deletion. After assignment of observations to tracks and track confirmation is performed, subsequent positions of the tracks are estimated.

2.3.3 Multiple Hypothesis Tracking

Multiple Hypothesis Tracking (MHT) algorithm is developed by Reid in 1979 [24] and several implementations of this algorithm are performed during the years up to

now. Cox and Leonard [29] propose a multiple hypothesis approach for building and maintaining a world model for an autonomous robot vehicle. Cox and Hingorani [20] present an efficient implementation of Reid's Multiple Hypothesis Tracking algorithm using the *k*-best hypotheses. They describe MHT as being the only statistical data association algorithm that integrates all the capabilities of track initiation, track termination, track continuation, explicit modeling of spurious measurements and explicit modeling of uniqueness constraints.

Multiple Hypothesis Tracking is an iterative algorithm. Instead of making observation-to-track assignments with the current data, this method generates hypotheses for the solution of the assignment problem. The iteration begins with some hypotheses for the solution, then for each hypothesis, each track's position in the following frame is predicted. The predicted position of the track is used to calculate the field of observation for the following frame. The distance between the track and the observations in its field of observation is taken into account to evaluate the hypotheses for that track. After the evaluation, some hypotheses are killed and the remaining is used to generate new hypotheses.

Blackman in [18] states that in problematic cases, like the one in Figure 2-1, MHT method generates alternative data association hypotheses and defers the association decision to the following frames. Hypotheses are propagated to the future and assignments are done using the future data, rather than deciding the assignment in the present frame. However, in GNN method the most likely assignment and in NN method the closest observation assignment is decided with the current data.

Multiple Hypothesis Tracking is chosen as the multiple target tracking method in this study.

CHAPTER 3

MOVING OBJECT DETECTION

The first part of visual surveillance systems is moving object detection. Moving object detection is the step at which the detection of the foreground objects is performed. Since the output of the moving object detection step is used as the input of the tracking step in surveillance systems, the accuracy of the moving object detection affects the overall performance of the system. The flow diagram of moving object detection algorithm is given in Figure 3-1.

Moving object detection algorithm consists of two basic steps which are foreground segmentation and sub-operations. In foreground segmentation step, regions corresponding to the moving objects are identified. This step is followed by sub-operations step in which shadows and noise are removed and components are labeled. Sub-operations consist of shadow removal, morphological operations and connected component labeling.

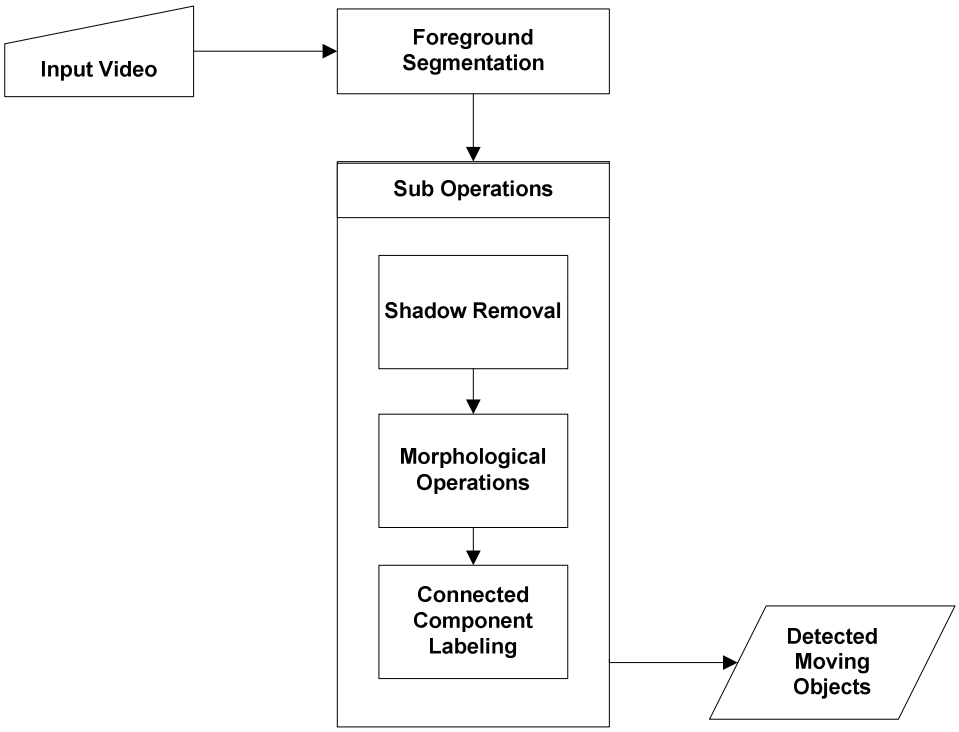


Figure 3-1 Flow diagram of moving object detection algorithm

3.1 Foreground Segmentation

Foreground segmentation is mostly based on a comparison between the input frame and a certain background model. The different regions between the input and the model are labeled as foreground based on this comparison. If the background is static, simple frame differencing algorithms can be sufficient for foreground segmentation. However, for the case in which the background is dynamic, in other words there is a repetitive motion in the background such as swaying of tree branches, more complex modeling is needed.

One of the methods which have good performance even in dynamic background cases is the Mixture of Gaussians method. It is an adaptive background subtraction method which is proposed by Grimson and Stauffer [22]. The main idea is to model the intensity values of each background pixel as the mixture of Gaussians and then compare them with the pixel values of the input frame to make the decision whether

the input pixel belongs to the background or not. The method proposed in [22] is robust to small motions in the background.

3.1.1 Mixture of Gaussians Method

Mixture of Gaussians method [22] is based on modeling every pixel as a mixture of Gaussians instead of using a single model for the entire background pixels. Gaussians corresponding to the background colors are identified by looking at the variance and persistence of each of the Gaussians. Pixels that do not fit the background distributions are considered foreground until there is a Gaussian that includes them with sufficient, consistent evidence supporting it to convert it to a new background mixture.

This method adapts itself to deal robustly with dynamic environment. The model is successful at detecting slowly moving objects since their color variance is greater than the background. Repetitive variations are also learned by the background model.

In Mixture of Gaussians method, the recent history of each pixel, $\{ X_1, ..., X_t \}$, is modeled by a mixture of *K* Gaussian distributions. The probability of observing the current pixel value is

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{3-1}$$

where *K* is the number of distributions, $\omega_{i,t}$ is an estimate of the weight (what portion of the data is accounted for by this Gaussian) of the i^{th} Gaussian in the mixture at time *t*, $\mu_{i,t}$ is the mean value of the i^{th} Gaussian in the mixture at time *t*, $\Sigma_{i,t}$ is the covariance matrix of the i^{th} Gaussian in the mixture at time *t*, and where η is a Gaussian probability density function and it is given in (3-2).

$$\eta(\mathrm{X}_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)} \tag{3-2}$$

K is determined by the available memory and computational power. Generally, K values from 3 to 5 are used. Also, for computational reasons, the covariance matrix is assumed to be of the form:

$$\sum_k = \sigma_k{}^2 I \tag{3-3}$$

This assumes that the red, green, and blue pixel values are independent and have the same variances. While this is certainly not the case, the assumption allows us to avoid a costly matrix inversion at the expense of some accuracy. Using a diagonal covariance would allow a Gaussian to represent that a particular channel showed more variation. Using a full covariance matrix would allow each Gaussian to model its local variation with more accuracy.

Thus, the distribution of recently observed values of each pixel in the scene is characterized by a mixture of Gaussians. Every new pixel of the current frame will be represented by one of the major components of the mixture model and used to update the model.

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. This threshold value is suitable for the cases with regions with different lightning. 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. The prior weight of the k^{th} Gaussian at time t is adjusted as follows:

$$w_{k,t} = (1-\alpha)w_{k,t-1} + \alpha M_{k,t} \tag{3-4}$$

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 matched ones are updated as follows:

$$\mu_t = (1-\varphi)\mu_{t+1} + \varphi x_t \tag{3-5}$$

$$\sigma_t^{\,2} = (1-\varphi)\sigma_{t-1}^{\,2} + \varphi(x_t - \mu_t)^T (x_t - \mu_t) \tag{3-6}$$

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

In order to decide what portion of the mixture model best represents background processes ω/σ ratios are considered. First, the Gaussians are ordered by the value of ω/σ. This value increases both as a distribution gains more evidence and as the variance decreases. After re-estimating the parameters of the mixture, it is sufficient to sort from the matched distribution towards the most probable background distribution, because only the matched models' relative value will have changed. This ordering of the model is effectively an ordered, open-ended list, where the most likely background distributions remain on top and the less probable transient background distributions gravitate towards the bottom and are eventually replaced by new distributions. Then the first *B* distributions are chosen as the background model, where

$$B = \arg\min_b \left(\sum_{k=1}^{b} \omega_k > T\right) \tag{3-7}$$

where *T* is a measure of the minimum portion of the data that should be accounted for by the background. This takes the "best" distributions until a certain portion, *T*, of the recent data has been accounted for. If a small value for *T* is chosen, the background model is usually unimodal. If this is the case, using only the most probable distribution will save processing. If *T* is higher, a multi-modal distribution caused by a repetitive background motion (e.g., leaves on a tree, a flag in the wind, etc.) could result in more than one color being included in the background model. This results in a transparency effect which allows the background to accept two or more separate colors.

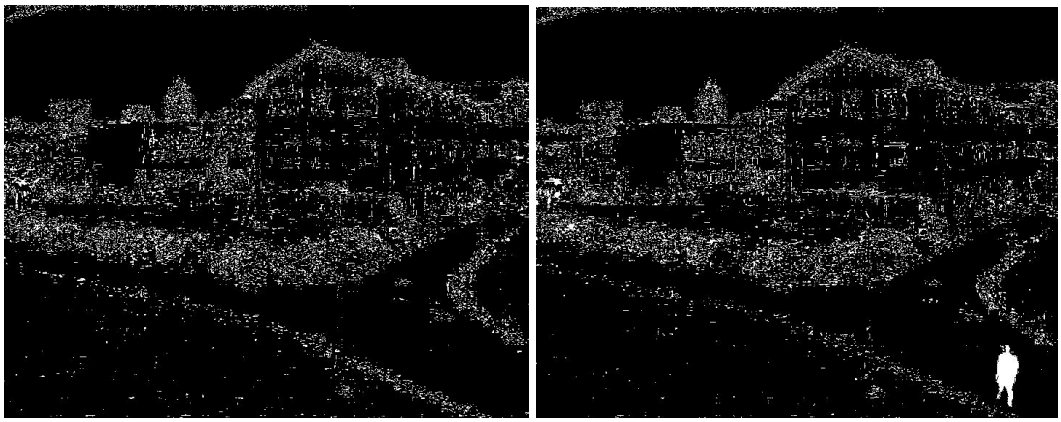
The algorithm used in this study to apply Mixture of Gaussians method is as follows:

1. Constant parameters such as the number of Gaussian components, the number of background components, positive deviation threshold, learning rate, foreground threshold and initial standard deviation are defined.
2. For each pixel in the frame, the initialization of the mean values for red, green and blue for each Gaussian are performed.
3. For each pixel in the frame, the initialization of the standard deviation values for red, green and blue for each Gaussian are performed.
4. For each Gaussian of each pixel in each frame, the difference between current pixel values and the means of the Gaussians is calculated.
5. For each Gaussian of each pixel in each frame, the calculated difference values are compared with the standard deviation values. If difference value is within the limits, corresponding Gaussian components are updated; if this condition is not satisfied then new Gaussian components are created.
6. For each Gaussian of each pixel in each frame, using the arithmetic mean of the standard deviations of red, green and blue, component rank for each Gaussian is calculated.
7. Calculated ranks are sorted.
8. Identification of foreground pixels is performed by checking whether a pixel value (the difference between current pixel value and the means of the Gaussians) is not within 2.5 standard deviations of a ranked Gaussian distribution.

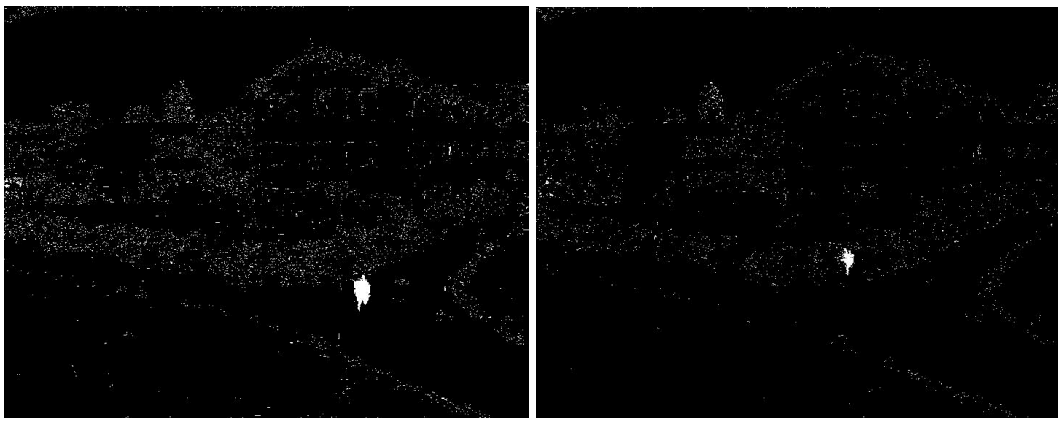
3.1.1.1 Results

Foreground segmentation using Mixture of Gaussians method is tested with PETS2001/Dataset1/Testing/Camera1 video [30] sequence and the results are given in Figure 3-2.

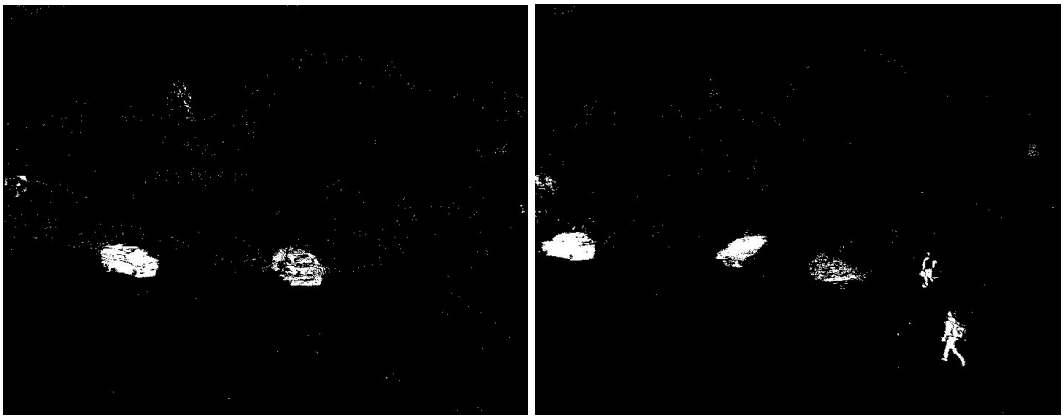
(a) (b)



(c) (d)



(e) (f)



(g) (h)

Figure 3-2 Results of foreground segmentation algorithm (a) 7th frame (b) 17th frame (c) 49th frame (d) 90th frame (e) 210th frame (f) 326th frame (g) 914th frame (h) 1321st frame

Consequently, simulation results show that Mixture of Gaussians method models background successfully. Being an adaptive background modeling method makes Mixture of Gaussians suitable for dynamic environments such as swaying of tree branches and robust to lightning changes.

3.2 Sub-Operations

The output of the foreground segmentation process may include shadows of moving objects and some noise. Therefore, before moving object tracking algorithm some other operations are applied to the output of foreground segmentation algorithm to remove shadows and noise and label the detected objects. These operations are shadow removal, morphological and connected component labeling.

3.2.1 Shadow Removal

The output of foreground segmentation process includes shadows of the moving objects since the intensity values of shadows of moving objects are different from

the intensity values of background. The need for the removal of shadows from the detected parts has two basic reasons. The first reason is that shadows cause false segmentation of the objects in terms of their features such as shape and size. The second reason is occlusion may occur as a result of large shadows. Therefore for accurate detection of moving parts in video frames, shadow removal is very important.

Shadow detection method proposed in [16] uses a color model that separates brightness from chromaticity component. This method is based on the brightness distortion α_i and the chrominance distortion CD_i values of pixels. The values α_i and CD_i obtained are used to classify a pixel in four categories: foreground, background, shadowed background and highlighted background. The shadowed regions are identified as the regions with similar chromaticity, but lower brightness than the background model.

The brightness distortion is obtained minimizing the equation (3-8).

$$\phi(\alpha_i) = (I_i - \alpha_i E_i)^2 \tag{3-8}$$

The color distortion of a pixel i is given by the equation (3-9).

$$CD_i = \|I_i - \alpha_i E_i\| \tag{3-9}$$

In the training period, the background is modeled statistically and each pixel is modeled by a 4-tuple $< E_i, \delta_i, a_i, b_i >$ where E_i is the expected color value, δ_i is the standard deviation of color value, a_i is the variation of the brightness distortion, and b_i is the variation of the chromaticity distortion of the i^{th} pixel. The expected color value of pixel i is given by $E_i = [\mu R(i); \mu G(i); \mu B(i)]$ where $\mu R(i)$, $\mu G(i)$ and $\mu B(i)$ are the arithmetic means of the i^{th} pixel's red, green, blue values computed over N background frames. Here a_i and b_i are calculated as follows:

$$a_i = RMS(\alpha_i) = \sqrt{\frac{\sum_{i=0}^{N} (\alpha_i - 1)^2}{N}} \tag{3-10}$$

$$b_i = RMS(CD_i) = \sqrt{\frac{\sum_{i=0}^{N} (CD_i)^2}{N}} \tag{3-11}$$

Using the calculated pixel components and suitable thresholds, pixel classification is performed. This method classifies a given pixel into four categories. A pixel in the current image is

- Original background, if it has both brightness and chromaticity similar to those of the same pixel in the background image.
- Shaded background or shadow, if it has similar chromaticity but lower brightness than those of the same pixel in the background image.
- Highlighted background, if it has similar chromaticity but higher brightness than the background image.
- Moving foreground object, if the pixel has chromaticity different from the expected values in the background image.

$$C(i) = \begin{cases} Foreground: \quad \overline{CD}_i > \tau_{CD} \quad or \quad \overline{\alpha}_i < \tau_{\alpha lo}, \quad else \\ Background: \quad \overline{\alpha}_i < \tau_{\alpha 1} \quad and \quad \overline{\alpha}_i > \tau_{\alpha 2}, \quad else \\ Shadowed \quad background: \quad \overline{\alpha}_i < 0, \quad else \\ Highlighted \quad background: \quad otherwise \end{cases} \tag{3-12}$$

The method proposed in [16] is an overall method that detects both moving parts and shadows. In our case, since Mixture of Gaussians (MoG) method is used for foreground subtraction, we use the proposed algorithm after MoG in order to identify the shadowed regions that are detected by MoG. Therefore, in our study, the method proposed in [16] is modified and applied using the following algorithm:

1) Obtain expected color values for red, green and blue for each pixel; these are the arithmetic means of the i^{th} pixel's red, green, blue values computed over N background frames.

2) Obtain standard deviation values for red, green and blue for each pixel; these are the standard deviation of the i^{th} pixel's red, green, blue values computed over N frame of the background frames.

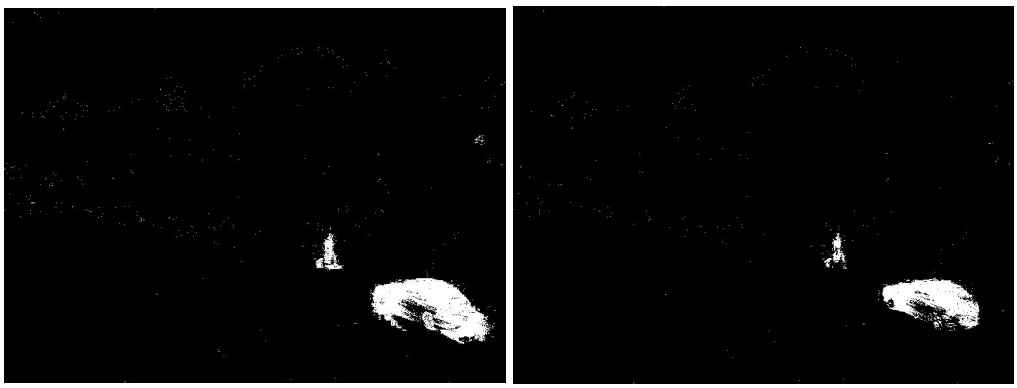
3) Calculate the variation of the brightness distortion for each pixel.

4) Calculate the variation of the chromaticity distortion for each pixel.

5) Construct the mask.

6) Using the mask, classify each pixel as foreground, background, shadowed background or highlighted background.

Shadow removal using the proposed method is tested with PETS2001/Dataset1/Testing/Camera1 video [30] sequence and the results are given Figure 3-3.



(a) (b)

Figure 3-3 Result of shadow removal algorithm (a) before shadow removal (b) after shadow removal

3.2.2 Morphological Operations

Morphological operations are image processing operations that process images based on shapes and they are used to remove the noise in the image which is obtained by moving object segmentation process. Mostly, morphological operations work mostly on binary images and apply structuring elements to the binary input

image, creating an output image of the same size. In most of the cases, the structuring element is sized $n \times n$ and has its origin at the center pixel. It is shifted over the input 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 operator, the pixel underneath the origin of the structuring element is set to a pre-defined value. Dilation and erosion are basic operations and can be combined into more complex sequences. The combination of erosion and dilation constitutes new operations called opening and closing. They are the most useful morphological filtering operations.

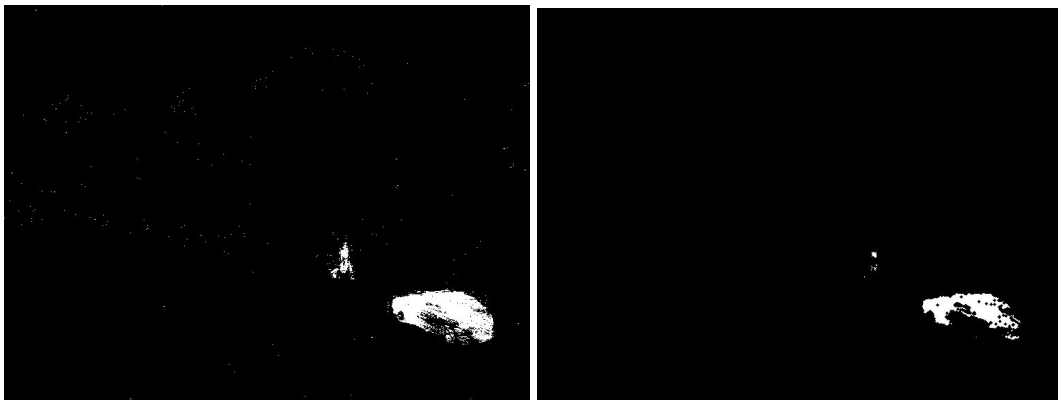
3.2.2.1 Erosion

Erosion operation shrinks foreground objects by etching away in other words eroding their boundaries. Individual pixels in the foreground image are removed by erosion using a structuring element in (3-13):

$$SE_{ersion} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3-13}$$

The structuring element is shifted over the image, if the origin of it coincides with a background pixel in the image, no operation is performed. If the origin of the structuring element coincides with a foreground pixel in the image and any of the surrounding '1' valued pixels in the structuring element coincides with a background pixel then the value of the pixel in the image is changed to a '0'.

Shadow removal using the proposed method is tested with PETS2001/Dataset1/Testing/Camera1 video [30] sequence and the results are given Figure 3-4.



(a) (b)

Figure 3-4 Result of erosion operation (a) before erosion (b) after erosion

3.2.2.2 Dilation

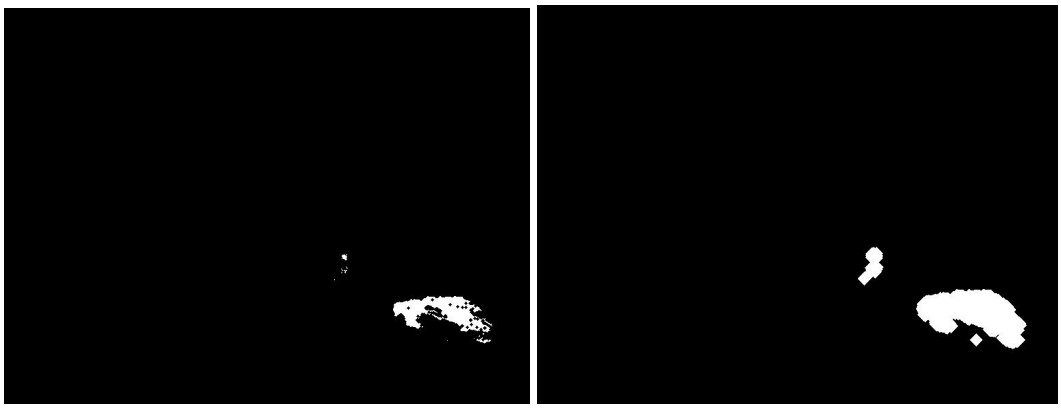
The dual operation of erosion is dilation. Dilation allows objects to expand, fills in small holes and connects disjoint object. A frequently used structuring element is given in (3-14).

$$SE_{dilation} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3-14}$$

The structuring element is slided across the image, if the origin of it coincides with a background pixel in the image, no operation is performed. If the origin of the structuring element coincides with a foreground pixel, “or” logic operation is performed on all pixels within the structuring element.

In order to remove the noise in the image, erosion and dilation can be applied to the binary image more than once and in different orders. In this study, after two erosion operations, three dilation operations are performed on the image.

Shadow removal using the proposed method is tested with PETS2001/Dataset1/Testing/Camera1 video [30] sequence and the results are given Figure 3-5.



(a) (b)

Figure 3-5 Result of dilation operation (a) before dilation (b) after dilation

3.2.3 Connected Component Labeling

After morphological operations, connected component labeling is applied to the foreground image. Connected component labeling identifies the objects in the binary image by finding the connected components in the image and labeling them.

CHAPTER 4

MOVING OBJECT TRACKING

Moving object tracking is one of the most important parts of visual surveillance systems. Moving object detection step is followed by moving object tracking which is based on the association of observations with tracks. Here, observations are the detected moving objects and the association is performed comparing the features of observations in the current frame with the features of tracks in the previous frames. The success of a surveillance system strongly depends on the accuracy of the observation-to-track association.

In this study, Multiple Hypothesis Tracking is used as the moving object tracking method. The method is evaluated using PETS2001 [30] and PETS2006 [31] data sets.

4.1 Multiple Hypothesis Tracking

Multiple Hypothesis Tracking is an iterative tracking method. Instead of making observation-to-track assignments with the current data, this method generates hypothesis for the solution of this assignment problem [18]. The iteration begins with some hypothesis for the solution, then for each hypothesis, each track's position is predicted for the following frame. The predicted position of the track is used to calculate the field of observation for the next frame. Hypotheses are evaluated using the distances between the tracks and the observations in their fields of observation. According to the evaluation, some hypotheses are killed and the remaining is used to generate new hypotheses.

In problematic cases, MHT method generates alternative data association hypotheses and postpones the decision to the forthcoming frames. Rather than deciding the assignment in the present frame, hypotheses are carried to the future and assignments are done using the future data. However, in GNN method the most likely assignment and in NN method the closest observation assignment is decided with the current data.

Reid [24] suggests an algorithm for MHT method, this algorithm consists of four major parts which are Cluster Formation (CLUSTR), Hypothesis Generation (HYPGEN), Reduction (REDUCE) and Mash. The flow diagram of his algorithm is given in Figure 4-1.

In the CLUSTER step of the algorithm, measurements are associated with previous clusters. The entire measurements and targets are divided into independent groups which are called clusters. By this way, in parallel, measurement-to-target association is considered in each cluster independently. A new target that does not belong to existing clusters forms a new cluster. A measurement that belongs to more than one previously independent cluster combines these clusters and the resulting cluster is called as a super cluster. The measurements and targets of the super cluster are the sum of the measurements and targets that were previously belonging to the cluster that forms the super cluster.

In the HYPOTHESIS GENERATION step, in each cluster, measurement-to-track association hypotheses are formed. The probability of each hypothesis is calculated and target predictions are updated.

The measurement oriented hypothesis generation technique is used as the hypothesis generation technique in which for each measurement every possible track is listed. While doing this, some rules are considered and these are; the existence of tentative tracks should be implied by previous related hypothesis, a track can be associated with only one measurement in one data set and a track can only be associated with a measurement if the measurement is in the gate of this track.

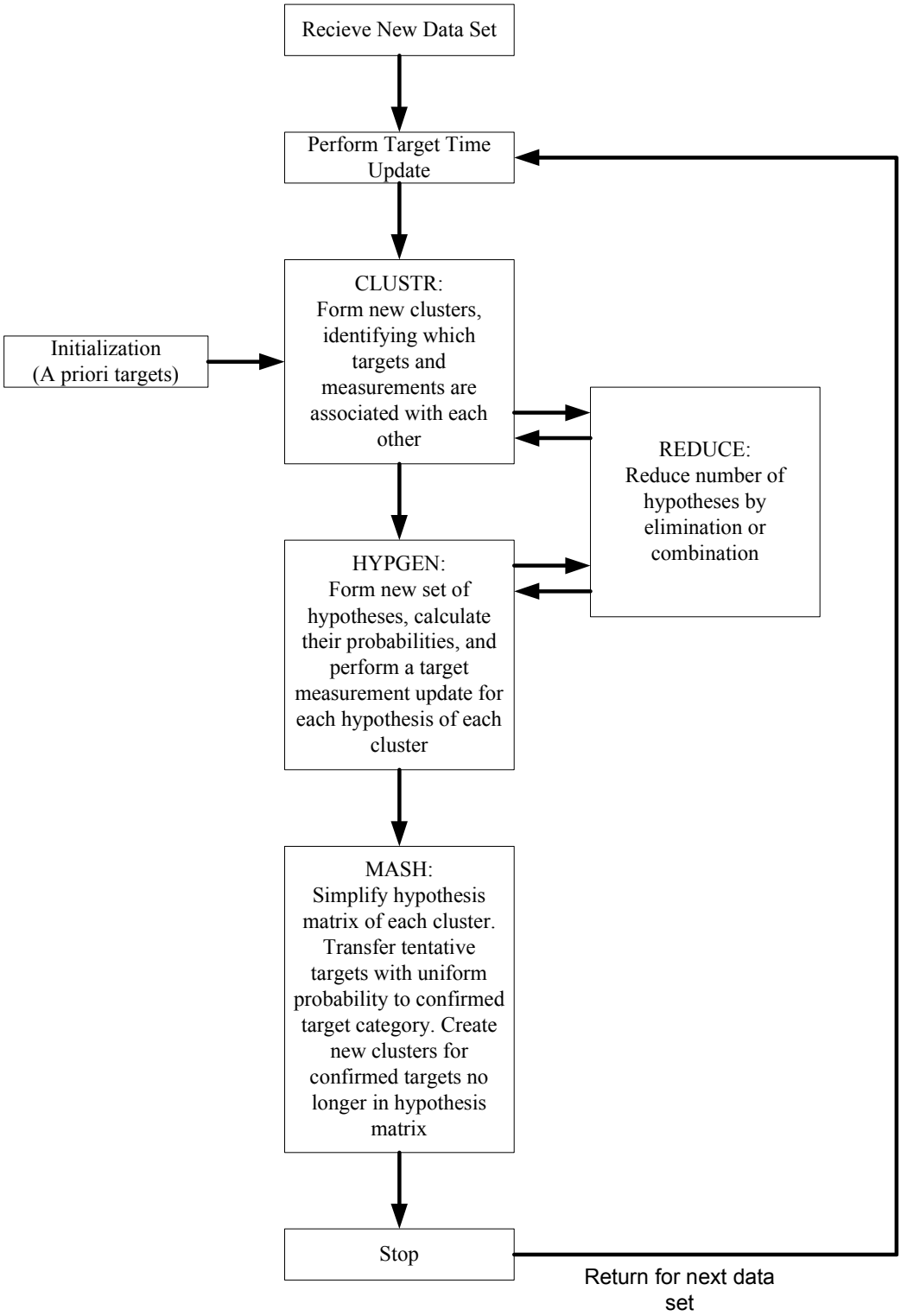


Figure 4-1 Basic flow diagram of Reid's algorithm for MHT, [24]

In the REDUCE part of the algorithm, low probability hypotheses are eliminated and similar hypotheses are combined.

In the MASH part, the hypothesis matrix is simplified. High probability tentative targets are moved to the confirmed target category and new clusters are formed for these targets.

4.1.1 Hypothesis-Oriented MHT

In hypothesis-oriented MHT implementations, measurement-to-track association hypotheses are carried from one scan to the next. Therefore, [18] each hypothesis carried from the previous scan causes an increase in the number of hypothesis and most of these hypotheses may be eliminated later because of their low scores (probabilities). Reid's algorithm is hypothesis-oriented and the potential explosion of new hypotheses that may result from an expansion of the old hypotheses has been a barrier to the practical implementation of Reid's algorithm. Cox et al. [20] proposes an efficient implementation of Reid's algorithm that only generates "good" hypotheses. Cox et al. use Murty's method [35] for finding the *m-best* solutions to the assignment problem in [20]. Using this approach, given $m_p(k - 1)$ hypotheses from the previous scan, the number of hypotheses formed in the current scan can be limited to $m(k)$ when m is an input parameter that could be set a priori or, presumably, could be chosen adaptively [18]. The important principle is that the generation of many low probability hypotheses, which resulted from earlier implementations of Reid's algorithm, is avoided.

4.1.2 Track-Oriented MHT

In track-oriented MHT implementations, in each scan, hypotheses are re-evaluated using the updated tracks. [18] Rather than maintaining, and expanding, hypotheses from scan to scan, the track-oriented approach discards the hypotheses formed on scan $k - 1$. The tracks that survive pruning are predicted to the next scan k where new tracks are formed, using the new observations, and reformed into hypotheses. Except for the necessity to delete some tracks based upon low probability, no

information is lost because the track scores, that are maintained, contain all the relevant statistical data.

The disadvantage of hypothesis-oriented approach is the explosion in the number of hypotheses generated in each scan and carried over to the future scans. Despite its disadvantage, in this study hypothesis-oriented approach is used, since maintaining hypotheses gives overall information about all associations. However, track-oriented approach may result in information loss. In track deletion, using hypothesis-oriented approach and therefore, maintaining hypotheses is very helpful since the tracks to be deleted can easily be understood as all the relation between hypotheses and tracks are kept.

4.2 Multiple Hypothesis Tracking

In this study, a multiple hypothesis tracking algorithm is applied to the objects detected in the moving object detection part of the system. The inputs of this algorithm are those detected objects and in the algorithm these detected objects are called as the observations. The observations are evaluated in each frame in terms of their position, size and color to achieve observation-to-track association in each frame of the input video.

The algorithm used for the application of MHT method consists of seven basic parts and these are Distance Matrix Calculation, Cluster Formation, Hypothesis Generation, Hypothesis Score Calculation, Track Update, Tree Management, Track Confirmation and Occlusion Handling. Flow diagram of the MHT method algorithm used in this study is given in Figure 4-2.

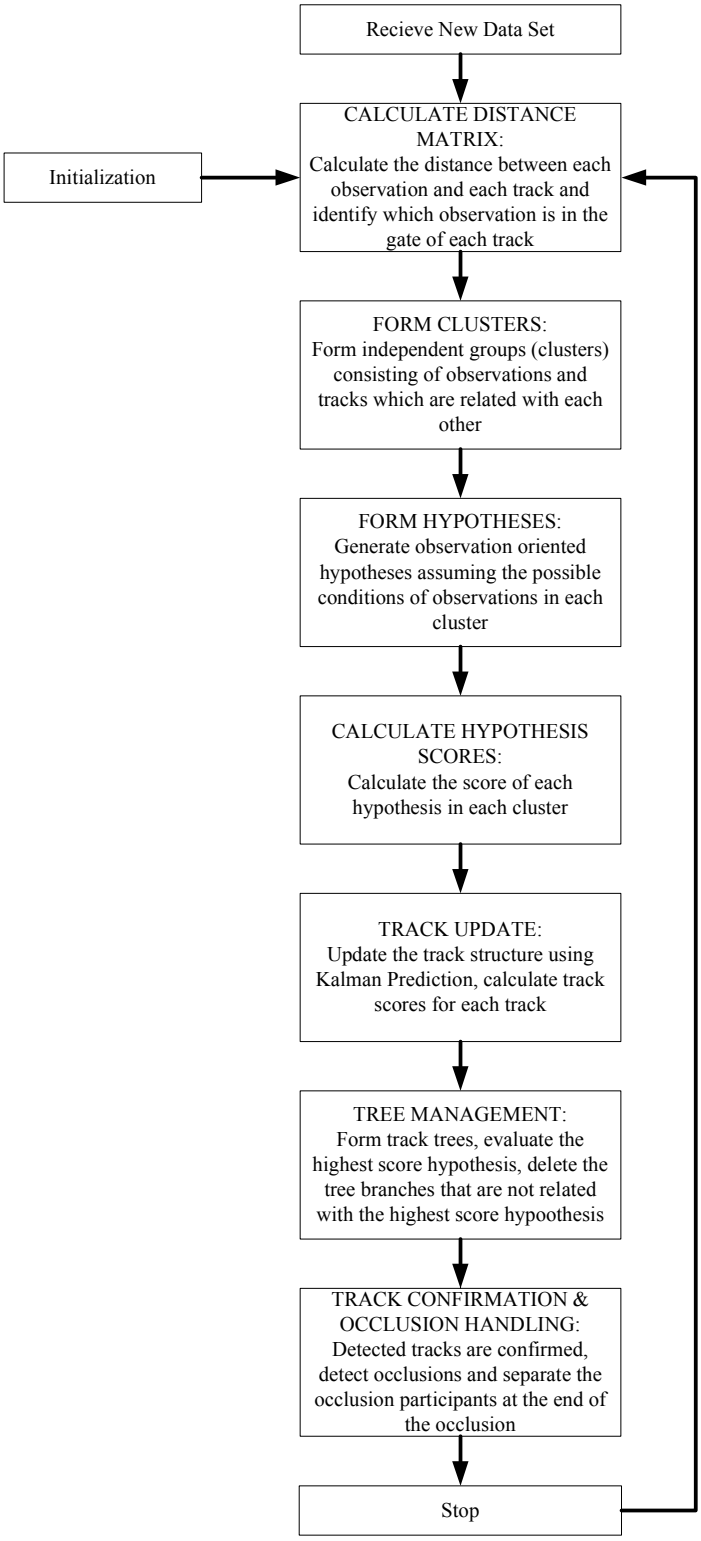


Figure 4-2 Flow diagram of the algorithm used in this study to apply the MHT method

Throughout the algorithm the information of each track is stored in a structure array called track as shown in Table 4-1.

Table 4-1 Track structure of n^{th} track

Gate width
Coordinate
Observation
Delete number
Confirmation number
Initialization
Covariance
State
Track number
Score
Cluster
Manage group
Size
Color
Window

In each frame, the values in the fields of the track structure are updated. The explanations of the fields of the array are:

- Gate width: The gate width of the track, the predicted area in which the track can appear, is stored in this field. Initially a predefined value, then in each frame the predicted gate width value (using Kalman prediction) is assigned as the gate width.
- Coordinate: The position information of the centroid of the track in x-y coordinates is stored in this field and it is a vector in the form of [x y]. Its initial value is the position vector of one of the observations, since in the first frame each observation is a track candidate. Then in each frame the corrected position vector (using Kalman prediction) is assigned as the coordinate value.
- Observation: The indices of observations that are in the gate of the track are stored in this field. Its initial value is an observation index, since in the first frame each observation is a track candidate.
- Delete number: The number of the consecutive frames that the track is not updated with an observation. Its initial value is zero. If this value reaches a predefined deletion number, this track is deleted.
- Confirmation number: The number of the consecutive frames that a tentative track is updated with an observation. Its initial value is zero. If this value reaches a predefined confirmation number, this track becomes a confirmed track, no more a tentative track.
- Initialization: The value of this field defines the state of the track. If the track is in the initialization state, this value is set to 1. Its initial value is zero.
- Covariance: The covariance value is stored in this field. It is the zero matrix initially and then in each frame, the covariance matrix calculated using Kalman prediction is assigned to this field.
- State: The state vector is stored in this field. Initially it is the zero vector and then in each frame the state vector calculated using Kalman prediction is

assigned. The state vector consists of the predicted x-y position, displacement in x-y coordinates differences, size, mean value of red, mean value of green and mean value of blue; $[x, y, \Delta x, \Delta y, S, \mu_R, \mu_G, \mu_B]$.

- Track number: The existing track number of the track is stored in this field. Its initial value is an observation index, since in the first frame each observation is a track candidate.
- Score: The calculated score value of the track is stored in this field. Its initial value is zero and in each frame it is calculated from the scores of the hypothesis in which the track takes place.
- Cluster: The cluster index in which the track takes place is stored in this field. Its initial value is an observation index, since in the first frame each observation forms a cluster.
- Manage group: The manage group index in which the track takes place is stored in this field. Its initial value is an observation index, since in the first frame each observation forms a manage group.
- Size: The size, total number of pixels, of the track is stored in this field. Its initial value is the size of an observation, since in the first frame each observation is a track candidate.
- Color: The RGB color means of the track is stored in this field. It is a vector in the form of $[\mu_R, \mu_G, \mu_B]$. Its initial value is the RGB color means of an observation, since in the first frame each observation is a track candidate.
- Window: The width and length value of the window that surrounds the track is stored in this field. It is a vector in the form of $[width, length]$, where *width* corresponds to x and *length* corresponds to y coordinates. Its initial value is the width and length values of the window that surrounds an observation, since in the first frame each observation is a track candidate.

4.2.1 Distance Matrix Calculation

Distance matrix consists of the distance information between each observation and each track. The distance information is important because the decision whether an

observation is related with a track or not is made considering the distance between the observation and the track. If this distance is smaller than the width of the track gate then, this observation is considered as related with that track. In the remaining part of the algorithm, this relation is examined to decide whether to update the track with this observation or not. Similarly, if the distance is greater than the width of the track gate then, this observation is considered as unrelated with that track.

Here, track gate is the predicted area in which the track can appear in the following frame of the video. Track gate is calculated (predicted) in each frame for each track with Kalman prediction. If a track does not have any observation within its track gate, the width of its gate is incremented according to Kalman prediction or if a track is updated with an observation in its gate than the track gate is decremented via Kalman prediction.

An example case is given in Figure 4-3. Here, P1 and P2 denote the predicted positions of the tracks T1 and T2, respectively. O1 and O2 denote the positions of two observations. O1 is in the track gate of P1 whereas O2 is in the gates of both tracks.

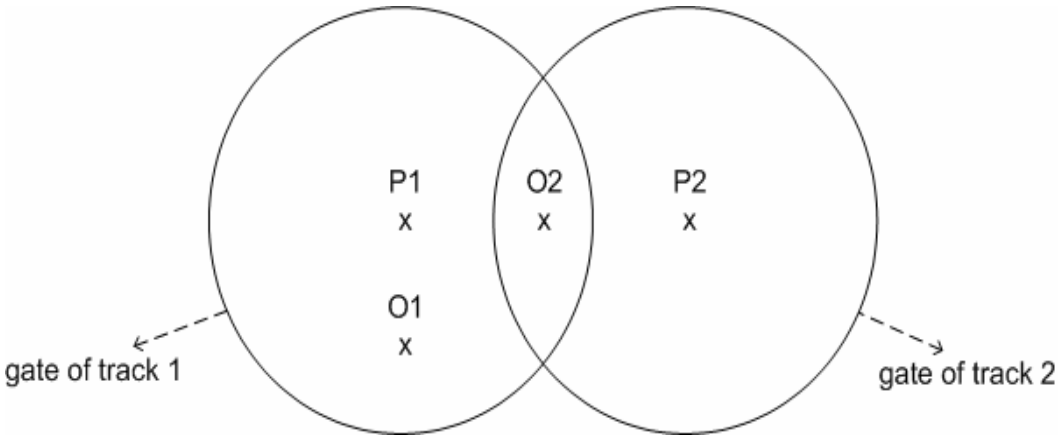


Figure 4-3 Example case with two existing tracks and two observations

Distance matrix is an *mxn* matrix where *m* is the number of observations and *n* is the number of tracks. In order to obtain the distance matrix, position information of observations and tracks are used. The position information consists of x-y

coordinate values. The distance is the Euclidean distance between each observation and each track and it is calculated as the square root of the sum of the squares of x and y coordinates differences between two points. If the Euclidean distance is smaller than the width of the track gate then, this observation is considered as related with that track. This relation is shown in (4-1) and detailed below.

$$d_{i,j} = \sqrt{\left(x_t - x_{o_i}\right)^2 + \left(y_t - y_{o_i}\right)^2} < r_{gate} \tag{4-1}$$

To do this, for each track,

- An *(mx1)* matrix is formed whose elements are the same representing the x-y coordinates of that track, $\left(x_t, y_t\right)$,
- An *(mx1)* matrix is formed whose elements represent the x-y positions of *m* observations, $\left(x_{o_i}, y_{o_i}\right)$,
- Second matrix is subtracted from the first one, $\left(x_t - x_{o_i}\right), \left(y_t - y_{o_i}\right)$,
- The square root of sum of the squares of these differences is taken in terms of x-y coordinates as in (4-2).

$$d_{i,j} = \sqrt{\left(x_t - x_{o_i}\right)^2 + \left(y_t - y_{o_i}\right)^2} \tag{4-2}$$

This value, the distance between the observations and the track, is compared with the gate width of that track. If the distance is greater, a relatively large number (it is chosen as 100), is assigned instead of that distance value. Then this number designates an observation that is outside the gate of this track.

After these calculations, a distance matrix, is formed whose $(i, j)^{th}$ element, $d_{i,j}$, is the distance between the i^{th} observation and j^{th} track if the i^{th} observation is in the gate of the j^{th} track and 100 otherwise.

The distance matrix of the example is given in Table 4-2. Here, the gate widths of the tracks T1 and T2 are both chosen as 20. $(1,2)^{th}$ element is 100 since O1 is not in

the gate of T2 and other elements of the matrix represent the corresponding distance values.

Table 4-2 Distance matrix for the example case

Observation\Track	T1	T2
O1	5	100
O2	10	10

In addition to distance matrix, logical distance matrix, is formed whose elements are either 1 or 0. This matrix is obtained by assigning 0 instead of 100 and 1 instead of the remaining elements of the distance matrix. In other words, if $(i, j)^{th}$ element of this matrix has the value 1 then, i^{th} observation is in the track gate of the j^{th} track. The logical distance matrix of the example is given in Table 4-3. $(1,1)^{th}$ element is 1 since O1 is in the gate of T1, $(1,2)^{th}$ element is 0 since O1 is not in the gate of T2, $(2,1)^{th}$ and $(2,2)^{th}$ elements are both 1, since O2 is in the gate of both T1 and T2.

Table 4-3 Logical distance matrix for the example case

Observation\Track	T1	T2
O1	1	0
O2	1	1

In this step, another matrix called distance score matrix is also calculated which will be used in hypothesis score calculation step. This matrix is obtained by assigning 0 instead of 100 in the distance matrix and divide the resulting non-zero elements,

$d_{i,j}^2$ by the corresponding track gate width, gw_j^2. Then the non-zero elements represent the normalized distance values by the gate width of the corresponding track, $d_{i,j} / gw_j$. As a result, the distance score matrix stores the score-distance relation between the observations and the tracks. If the distance is large, the score is low because of the inverse ratio.

4.2.2 Cluster Formation

Observations and related tracks have to be divided into independent groups in order to be able to examine them separately. While forming these independent groups which are called as clusters, the directly and indirectly related observations and tracks are held in the same group. The relationship information comes from the logical distance matrix. Here, direct and indirect relationship can be explained with an example. Say, observations O1 and O2 are related with track T1 and observations O2 and O3 are related with track T2. Then (O1, O2, T1) and (O2, O3, T2) are directly related whereas, because of O2 (O1, O2, O3, T1, T2) are indirectly related.

Clusters are stored in a *2xn* cell array where *n* denotes the number of clusters. The first row of the cell array consists of the track indices and the second row consists of the observation indices in that cluster. Cluster cell array format is shown in Table 4-4.

Table 4-4 Cluster cell array format

Track & Obs. \ Cluster	C1	C2	…	Cn
Track Field	…	…	…	…
Observation Field	…	…	…	…

While forming clusters logical distance matrix is considered and the following rules are used:

1. An observation that is not related with any track forms a cluster itself.
2. A track that is not related with any observation forms a cluster itself.
3. Directly and indirectly related observations and tracks takes place in the same cluster.

The algorithm which is used to achieve these rules is given in Figure 4-4.

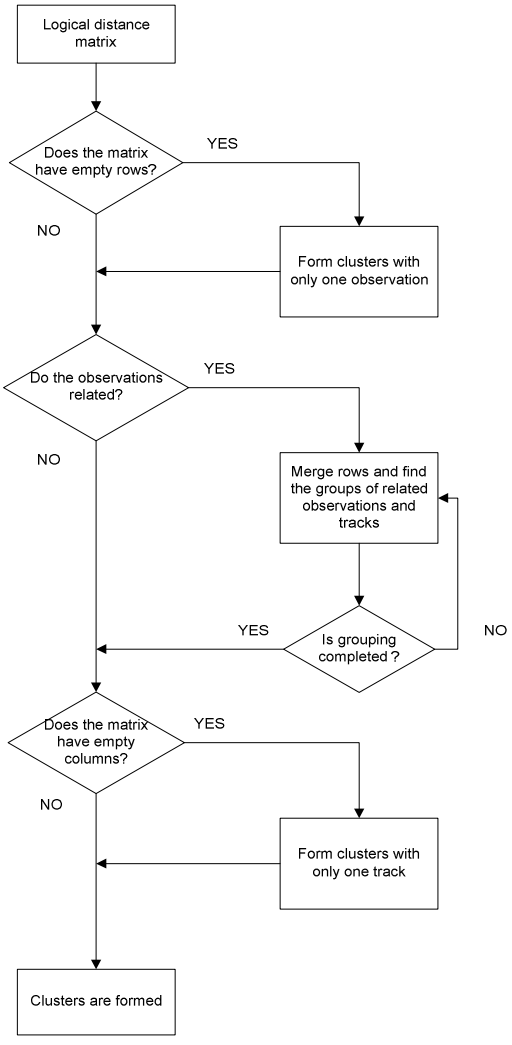


Figure 4-4 Cluster formation algorithm

The details of the algorithm are given below:

- Form Clusters With Only One Observation: An observation that is not related with any track forms a cluster itself. This cluster consists of only one element which is that observation. Such an observation is found from the distance matrix by looking at its empty rows. A row is called empty if none of its elements are 1. Then the indices of the empty rows of the logical distance matrix give the indices of the observations that are not related with any track. Each of these observations forms a cluster whose track index is empty. After examining empty rows, they are removed from the logical distance matrix.

- Check Whether the Observations are Related: The rows of the logical distance matrix are compared one by one. If two rows have 1 in the same column, the corresponding observations are related.

- Merge Rows and Find the Groups of Related Observations and Tracks: If the observations of two rows are related, these rows are merged, in other words logical OR operation is applied to these two rows. The resulting row is replaced with the upper row in the logical distance matrix and the lower one is erased. This step is applied to the example case as shown in Figure 4-5.

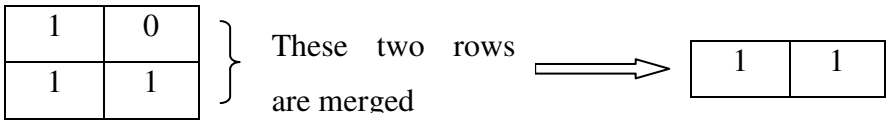


Figure 4-5 Row merging for cluster formation

- Check Whether Grouping is Completed: This rule is repeated until the columns have at most one 1 in its elements. To do this, the sum of the elements of a column is checked. If the sum is greater than 1, there still

exists rows having a 1 in the same column and the rule has to be repeated. This condition is checked in the case of the example as in Figure 4-6.

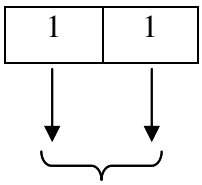


Figure 4-6 Checking the completion of grouping for cluster formation

Up to now in this step, the rows having a 1 in the same column are merged; in other words; the observations related with the same track are summed and placed in the same row.

While forming a cluster using the observation-track relationship from the resulting distance matrix, each row of this matrix is taken into consideration since each row of this matrix will form a cluster. The track indices of a cluster are found from the column indices of the 1's in a row of this matrix. The observation indices of the cluster is found from the row indices of the 1's in the column of the track in consideration in the original input logical distance matrix.

- Form Clusters With Only One Track: A track that is not related with any observation forms a cluster itself. This cluster consists of only one element which is that track. Such a track is found from the resulting distance matrix by looking at its empty columns. A column is called empty if none of its elements are 1. Then the indices of the empty columns of the resulting logical distance matrix give the indices of the tracks that are not related with any observation. Each of these tracks forms a cluster whose observation index is empty.

The cluster cell array for the example case is formed as given in Table 4-5. This table is formed assuming that there are some other tracks and observations that are not related with the tracks and observations considered in this example. Therefore those unrelated tracks and observations form other clusters. It is also assumed that the cluster formed for the example case is the second cluster.

Table 4-5 Cluster cell array for the case of the example

Track & Obs. \ Cluster	C1	C2	…	Cn
Track Field	…	T1,T2	…	…
Observation Field	…	O1,O2	…	…

4.2.3 Hypothesis Generation

For each cluster a hypothesis set is generated and these are stored in a *1xn* cell array called hypothesis matrix where *n* denotes the number of clusters. Each cell of the hypothesis matrix is an *mxk* matrix where *m* is the number of generated hypotheses for this cluster and *k* is the number of observations. Hypothesis cell array format is shown in Table 4-6.

Table 4-6 Hypothesis cell array format

Track & Obs. \ Cluster	C1	C2	…	Cn
Hypothesis Field	…	…	…	…

In this algorithm, observation oriented hypothesis generation technique is used. Observation oriented hypothesis generation technique is listing possible tracks for each observation. The rules used for hypothesis generation are as follows:

1. If a cluster does not contain any observation, its corresponding hypothesis matrix cell consists of only one element which is the index of the track in that cluster. And that means there is one possible hypothesis which tells the track in the cluster remains alive.

2. If a cluster does not contain any track, its corresponding hypothesis matrix cell consists of two kinds of hypotheses which are false alarm and new track.

   False alarm hypothesis means the observation in the cluster is detected wrongly therefore; this observation does not update or initiate any track. False alarm hypothesis is labeled with a zero.

   New track hypothesis means the observation in the cluster initiates a new track that does not exist before. New track hypothesis is labeled with new track index. And it is calculated by incrementing the maximum existing track index by one.

3. If a cluster contains both observations and tracks, three kinds of hypothesis are generated for each observation which are false alarm, new track and track update. Since this algorithm uses the observation oriented hypothesis generation technique, hypotheses are generated based on the possibilities of observations.

   First kind is false alarm hypothesis that means this observation does not update or initiate any track. This kind of hypothesis is labeled with a zero.

   The second type is the one in which a new track is initiated with that observation and this kind of hypothesis is shown with new track index and again this new track index is calculated by incrementing the maximum existing track index by one.

   The third type consists of the hypothesis of every possible association of each observation with tracks. Here the word “possible” is used in order to emphasize that an observation can only be associated with a track if and only if it is in the gate of that track. This type of hypothesis defines the

update of a track with an observation in its gate and it is shown with the existing track index.

In such a cluster containing both observations and tracks, hypothesis formation is as follows: For the first observation these three type hypotheses are formed and held in a vector called hypothesis vector containing one hypothesis in each row. This vector is the upper part of the first column of the hypothesis matrix. Each column of the hypothesis matrix is related with one observation. For the case of the example, the hypothesis of O1 and O2 are given in Figure 4-7 where 3 and 4 are the new track indices generated for O1 and O2 respectively.

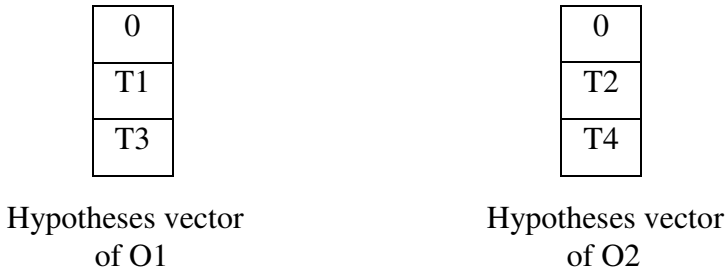


Figure 4-7 Hypotheses vector formation for the example case

The hypotheses of the second observation are formed like the first one and the hypothesis vector of this observation is generated in the same manner.

After this, the first observation's hypothesis vector is copied as the number of the hypothesis of the second observation (number of the elements of the second hypothesis vector) and the copies of the first vector are placed under the first vector. Then, in the second column of the hypothesis matrix, each element of the second observation's hypothesis vector is placed through the rows of each copy of the first observation's hypothesis vector.

For the third observation, the hypothesis vector is formed and the copying process is repeated as if the two columns are considered as the first observation's hypothesis vector. Therefore, these two columns are copied as

the number of the hypothesis in the third observation's hypothesis vector and the elements of the third observation's hypothesis vector are placed as in the second observation case. The hypothesis matrix of the example case is shown in Figure 4-8.

0	0
T1	0
T3	0
0	T1
T1	T1
T3	T1
0	T2
T1	T2
T3	T2
0	T4
T1	T4
T3	T4

Figure 4-8 Hypothesis matrix of the example case

This procedure is repeated for other observations and during the procedure; hypotheses of the examined observations are copied in the same manner in order to generate the hypothesis matrix. The rows of this matrix give the hypotheses. A simplification is performed to prevent the update of the same track with more than one observation in the same hypothesis. This means a track index cannot appear more than once in a row of the hypothesis matrix. Therefore, these rows that have repeated track indices as its elements are cleared. This rule is applied to the hypothesis matrix of the example case as in Figure 4-9.

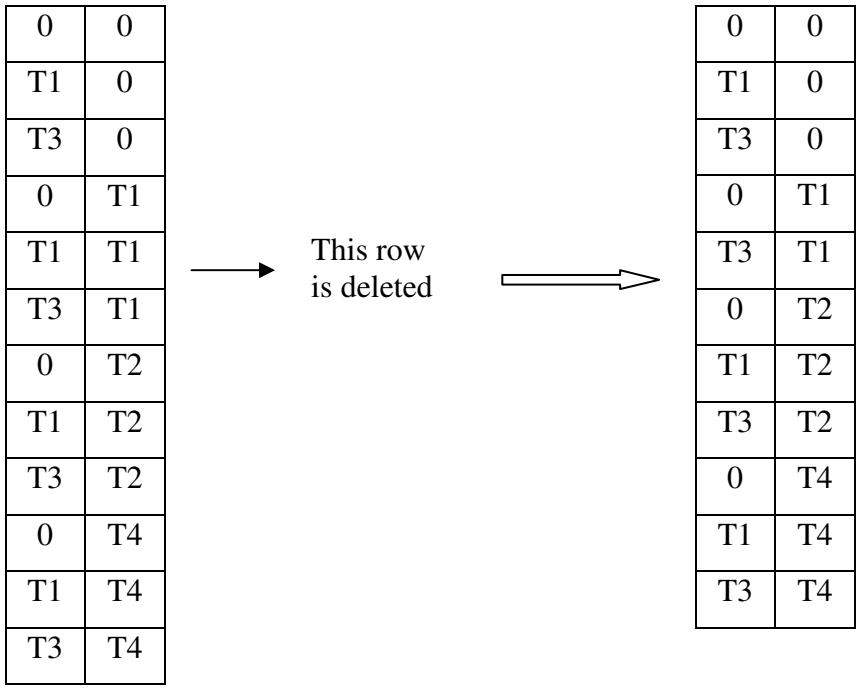


Figure 4-9 Hypothesis matrix row deletion for the example case

After repeating this process for each cluster, the output hypothesis matrix is generated which is a *1xn* cell array where *n* is the number of the clusters.

Applying these steps also satisfies some criteria implied in Reid's MHT Method; in a hypothesis, a track can be associated with only one observation and this observation has to be in the track gate.

4.2.4 Hypothesis Score Calculation

In this step, hypotheses scores are calculated and stored in a *1xn* cell array called score matrix where *n* denotes the number of clusters. Each cell of the cell array is a *1xm* vector where *m* is the number of generated hypotheses for this cluster.

If there is an observation-to-track association, in other words, a track is updated with an observation; the score of the hypothesis is calculated according to the features: distance, size, color, gate and track history. Therefore, the observation-to-track association is evaluated via these criteria and the resulting score shows the validity of that association.

Here, distance is the absolute difference of the x-y positions of the observation and the track. This information is obtained from the distance score matrix. Distance score matrix stores the distance-to-gate width ratio and it gives idea about how close the observation and the track are to each other; if the distance score is high, the distance is small and vice versa.

Similar to distance score, color and size scores give idea about how much the size and color of the observation match with the size and color of the track. Therefore, the reality of the association is examined in terms of their size and color information.

Gate and track history scores give idea about the previous associations of the track. When an observation-to-track association is performed, the next possible position of the track is predicted more precisely compared with a large track gate case, and then the track gate is small. Therefore, the smaller the track gate, the higher the score. Similarly, track history score is high if observation-to-track associations are performed in last scans.

The score of a hypothesis is calculated individually for each of these criteria. The total score is the weighted sum of the individual scores calculated via these features with normalized weights and this calculation is shown as follows:

$$\text{TotalScore} = \frac{\left(w_{\text{d}} \times D.S. + w_{\text{s}} \times S.S. + w_{\text{c}} \times C.S. + w_{\text{g}} \times G.S. + w_{\text{h}} \times H.S.\right)}{\left(w_{\text{d}} + \text{v}_{\text{s}} + w_{\text{c}} + w_{\text{g}} + w_{\text{h}}\right)} \tag{4-3}$$

$$0 \le w_c, w_s, w_g, w_h, w_d \le 1 \tag{4-4}$$

where w_d is the weight of the distance score (*D.S.*), w_s is the weight of the size score (*S.S.*), w_c the weight of the color score (*C.S.*), w_g is the weight of the gate score (*G.S.*) and w_h is the weight of the track history score (*H.S.*).

Hypothesis score calculation is performed in every cluster according to these rules:

- If the hypothesis index is 0, which means a false alarm, a predefined score value for "a false alarm" is assigned which is relatively small.
- If the hypothesis index is a new track index and the cluster contains old (previously existing) tracks, hypothesis score is the predefined score value of the case "new track found in a cluster that has some other tracks previously".
- If the hypothesis index is a new track index and the cluster does not contain old (previously existing) tracks, hypothesis score is the predefined score value of the case "new track found in a cluster that has no tracks previously".
- If the hypothesis index is an index of an existing track and the cluster does not contain any observations, hypothesis score is the predefined score of the case of "an existing track without an update because there is no observation to update this track".
- If the hypothesis index is an index of an existing track and the cluster contains observations, this means an existing track is updated with an observation in its track gate. Individual scores according to distance, size, color, gate and track history are calculated and then the total score of this observation-to-track association is obtained with the formula (4-1). This score gives idea about the validity of this association.

  Distance score calculation:

  Maximum distance score is 100 and this is the score of the case in which the center of the track gate coincides with the center of the observation. A predefined step score multiplied by the normalized distance (calculated in

the distance score matrix) is subtracted from 100 (maximum score) as shown in (4-5).

$$\text{Distance score} = 100 - (\text{distance}/\text{gate width}) * \text{min dist score diff} \quad (4\text{-}5)$$

Therefore, a distance score assigned for an observation becomes smaller for larger distances.

Size score calculation:

Maximum size score is 100 and this is the score of the case in which the size of the observation matches with the size of the track. A predefined step score difference multiplied by the ratio of the size difference to step size is subtracted from 100 (maximum score) as shown in (4-6).

$$\text{Size score} = 100 - (\text{size diff}/\text{step size}) * \text{step size score diff} \quad (4\text{-}6)$$

Therefore, a size score assigned for an observation becomes smaller for mismatched sizes.

Color score calculation:

Maximum color score is 100 and this is the score of the case in which the color of the observation matches with the color of the track as shown in (4-7). Arithmetic mean of red, green and blue is used in the calculation.

$$\text{Color score} = 100 - (\text{color diff}/\text{step color}) * \text{step color score diff} \quad (4\text{-}7)$$

A predefined step score multiplied by the color score is subtracted from 100 (maximum score). Therefore, a color score assigned for an observation becomes smaller for mismatched colors.

Gate score calculation:

Maximum gate score is 100 and this is the score of the case in which the gate of the track is small, in other words, the track is continuously updated with an observation. A predefined step score difference multiplied by the ratio of the gate difference to step gate is subtracted from 100 (maximum score) as shown in (4-8).

$$\text{Gate score} = 100 - (\text{gate diff} / \text{step gate}) * \text{step gate score diff} \quad (4\text{-}8)$$

Therefore, a gate score assigned for an observation becomes smaller for tracks that are not updated.

History score calculation:

Maximum history score is 100 and this is the score of the case in which the track is updated in a number of, say 5, consecutive frames. A predefined history score difference multiplied by the ratio of the history difference to step history is subtracted from 100 (maximum score) as shown in (4-9).

$$\text{Histroy score} = 100 - (\text{history diff} / \text{step history}) * \text{step history score diff} \quad (4\text{-}9)$$

Therefore, a history score assigned for an observation becomes smaller for tracks that are not updated.

4.2.5 Track Update

Track management consists of two steps which are Kalman Prediction and Track Score Calculation as shown in Figure 4-10.

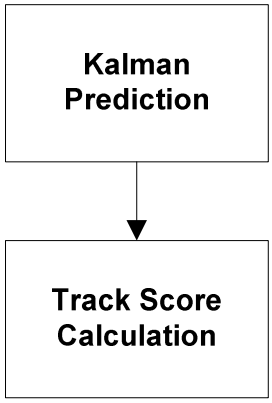


Figure 4-10 Flow diagram of track update

The update of track structure parameters is performed in Kalman prediction step. Then, in Track Score Calculation step, track scores are calculated for each track using the scores of the hypothesis in which that track takes place.

4.2.5.1 Kalman Prediction

Kalman filter [35] is used for track update, in other words, to predict the parameters in the track structure. Kalman filter is a set of mathematical equations that provides an efficient computational means to estimate the state of a process by minimizing the mean of the squared error. Using this filter past, present and future states of the system can be estimated without knowing the exact behavior of the system. For this reason Kalman filter is very popular since 1960 and used in this study for track update.

The Kalman filter addresses the general problem of trying to estimate the state $x \in R^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{4-10}$$

with a measurement defined by the equation

$$z_k = Hx_k + v_k \tag{4-11}$$

The random variables w_k and v_k represent the process and measurement noise, respectively. They are assumed to be independent of each other, white, and with normal probability distributions.

$$p(w) \sim N(0, Q) \tag{4-12}$$

$$p(v) \sim N(0, R) \tag{4-13}$$

In practice, the process noise covariance Q and measurement noise covariance R matrices might change with each time step or measurement, however assume they are assumed to be constant in our work.

The next state of a process is estimated by Kalman filter by the scheme given in Figure 4-11. The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time.

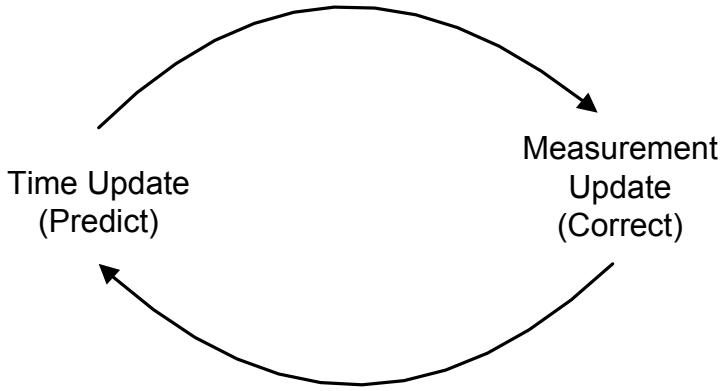


Figure 4-11 Kalman filter cycle

For time-update and measurements update steps of Kalman filter, there are equations by which a priori $(\hat{x}_k^-)$ and posteriori $(\hat{x}_k)$ estimates are determined.

Time update equations (predict):

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{4-14}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{4-15}$$

Measurement update (correct):

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{4-16}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{4-17}$$

$$P_k = (I - K_k H)P_k^- \tag{4-18}$$

The state vector of the track consists of the x-y position, displacement in x-y coordinates differences, size, mean value of red, mean value of green and mean value of blue; $[x, y, \Delta x, \Delta y, S, \mu_R, \mu_G, \mu_B]$. Kalman filter is performed for the prediction of the next state of this state vector using the following rules in this study:

- If the track is updated with an observation, then the state of this observation is used for prediction.

- If the track is not updated with an observation, then the previous state of the track is used for prediction.

Track gate width is calculated from the covariance of the Kalman filter and it is updated after each prediction. When the track is updated with an observation, the track gate width becomes smaller; similarly when the track is not updated, its gate width is enlarged.

4.2.6 Track Score Calculation

Track scores are calculated from hypothesis scores. The score of each track is the sum of the scores of the hypothesis in which the index of that track is contained. Track score calculation is performed using the following rules:

- Track score calculation of a new track: For each hypothesis in a cluster, new tracks are found by comparing the track indices with the existing track indices; the track indices that are greater than the existing track indices gives the new track indices. Then for each new track, the hypotheses are searched in which that track index is contained. The score of each new track is the sum of the scores of the hypothesis in which the index of that track is contained. The score of each track is stored in the track structure.
- Track score calculation of an existing track which is updated with an observation: These are the tracks from previous frames having one or more observation in their track gates. The score of these tracks, when they are updated with each of these observations, are calculated using the hypothesis scores matrix. Here, the column of the hypothesis score matrix which corresponds to the observation updating that track is considered. Then the score of each track is the sum of the scores of the hypothesis in which the index of that track is contained. The score of each track is stored in the track structure.
- Track score calculation of an existing track which is not updated with an observation: This situation can occur when there is no observation in the

gate of the track or the observation in the track gate initiates a new track or the observation in the track gate is a false alarm. The score of the track in one of these situations is the sum of the scores of the hypothesis in which the index of that track is not contained. The score of each track is stored in the track structure.

4.2.7 Track Tree Management

Track tree management consists of three steps which are Track Tree Formation, Highest Score Hypothesis Evaluation and Tree Pruning as shown in Figure 4-12.

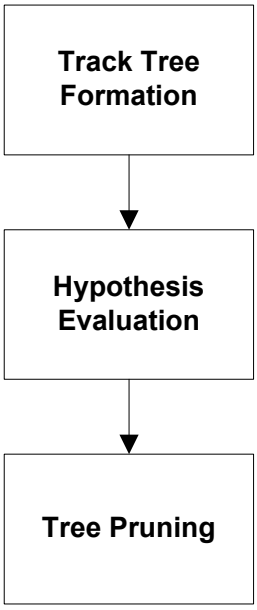


Figure 4-12 Flow diagram of track tree management

The trees which are the update structures of tracks with observations are formed in Tree Formation step, hypotheses are evaluated in Hypothesis Evaluation step and then according to the result of the Hypothesis Evaluation step, the tree braches that do not represent the highest score hypothesis are deleted in the Tree Pruning step.

4.2.7.1 Track Tree Formation

Track trees are the structures that are used for the implementation of the deferred decision logic of MHT. The history of the relationship between tracks and hypotheses are stored in track trees. Track trees are used for the selection of the most probable observation-to-track association and deletion of the associations contradicting with that most probable association.

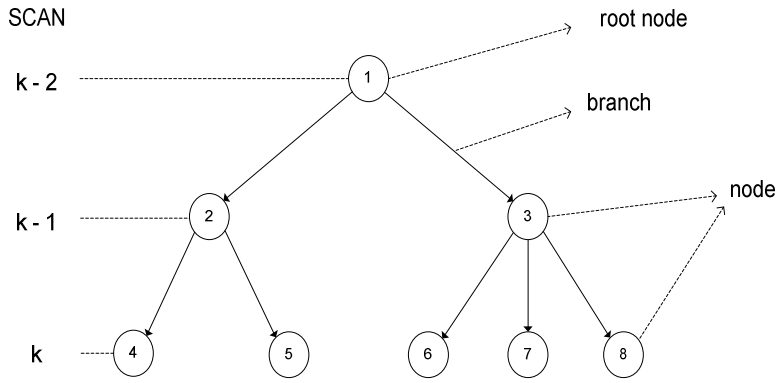


Figure 4-13 Track tree elements

A track tree consists of some elements such as root node, node, branch, decision depth and current depth as in Figure 4-13. A root node is the source node or the origin of a tree, in other words, it is the confirmed track from which all the track candidates are generated using hypotheses. Nodes are the track candidates that are generated with hypotheses and branches are the hypotheses that represent the observation-to-track association between nodes. The decision depth of a track is the number of scans that the association decision is deferred. If the decision depth is *k*, then the observation-to-track association decision is made considering the data of recent *k* scans or in other words, the decision is postponed for *k* scans. It is 3 in Figure 4-13. The current depth is the number of scans whose information is stored in that tree, or in other words, it is the number of recent scans that are included in the history of the tree.

The decision depth of the tree, or the maximum allowed depth of the tree, has an important effect on the performance of the MHT since the association decision is made considering that much earliest data. In this study, decision depth is chosen as 4 therefore, the decision of an association is made after 4 scans.

In this study, in order to build the track tree, a structure called manage track is formed for each tree as in Table 4-7.

Table 4-7 Manage track structure of n^{th} tree

Track indices
Track positions
Track groups
Track scores
Current depth
Tree number
Root node
History
Confirmation flag
Deletion flag
Confirmed list index

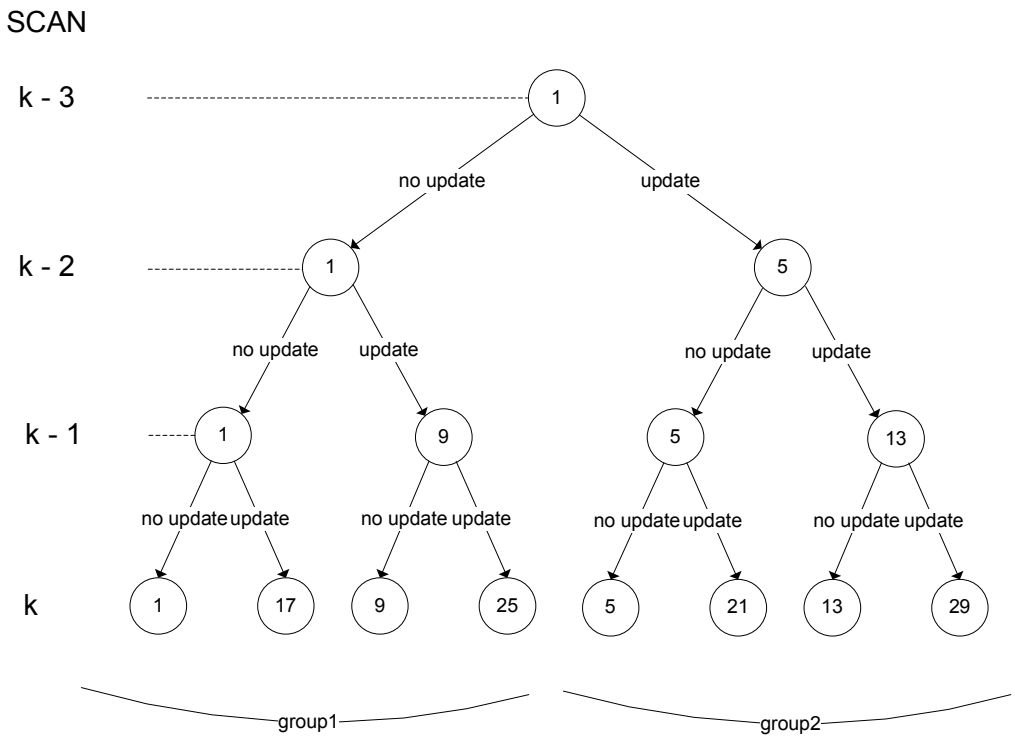


Figure 4-14 Example track tree

In each frame the values in the fields of the manage track structure is updated. An example track tree is given in Figure 4-14. The explanations of the fields of the array are:

Track indices: The indices of the tracks which are the nodes of the tree are stored in this field. Since most of the trees have many tracks and the depth of the tree is greater than 1, these indices are also held in a matrix called track indices as in Table 4-8.

Table 4-8 Track indices matrix for the example track tree

Node at depth 1 (root node)	Nodes at depth 2	Nodes at depth 3	Nodes at depth 4
1	1	1	1
0	5	9	17
0	0	5	5
0	0	13	21
0	0	0	9
0	0	0	25
0	0	0	13
0	0	0	29

Track positions: The branching information is stored in this field as a matrix called track positions. If a node branches into two nodes in the following frame, the corresponding cell value is 2. An example of this matrix is shown in Table 4-9. Each branch designates a hypothesis; this means if a cell value is 2, there are 2 hypotheses for this node. The first hypothesis is the observation is false alarm therefore the track is not updated and the second one is the association of the track with an observation and the track is updated with an observation (this observation index is stored in that track's track structure). In fact, there is one more hypothesis that is generated for an observation and it is the initiation of a new track. However, since the tree is built concerning the track's point of view, this hypothesis is not shown as a third hypothesis in the track positions matrix; instead it is represented as a new root node.

Table 4-9 Track positions matrix for the example track tree

Node at depth 1 (root node)	Nodes at depth 2	Nodes at depth 3	Nodes at depth 4
2	2	2	0
0	2	2	0
0	0	2	0
0	0	2	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Track groups: The groups of the tracks which are the nodes of the tree are stored in this field. These groups are also held in a matrix called track groups as in Table 4-10. The group number is either 1 or 2 since there are always 2 groups in a tree. The first group is the group of track nodes which are originated from the no update case of the root node, in other words, these tracks are generated from the hypothesis of the root node is not updated with an observation. The second group is the group of track nodes which are originated from the update case of the root node. This notation is necessary for the tree pruning step which will be discussed later since the decision of association is made after k scans. By this way, it is easy to understand which tracks are going to be deleted when the decision is the update or no update of the root node track.

Table 4-10 Track groups matrix for the example track tree

Node at depth 1 (root node)	Nodes at depth 2	Nodes at depth 3	Nodes at depth 4
0	1	1	1
0	2	1	1
0	0	2	2
0	0	2	2
0	0	0	1
0	0	0	1
0	0	0	2
0	0	0	2

Track scores: The scores of the tracks which are the nodes of the tree are stored in this field. These scores are also held in a matrix called track scores as in Table 4-11.

Table 4-11 Track scores matrix for the example track tree

Node at depth 1 (root node)	Nodes at depth 2	Nodes at depth 3	Nodes at depth 4
0.1407	0.8525	0.8579	0.8605
0	0.1475	0.1421	0.1408
0	0	0.8515	0.8530
0	0	0.1485	0.1467
0	0	0	0.6123
0	0	0	0.2842
0	0	0	0.5478
0	0	0	0.1355

Current depth: It is the current depth of the track tree which is the number of scans that are included in the history of the tree.

Tree number: Track trees are numerated and this field contains the index given for that track tree.

Root node: It is the track index of the root node and root node is, as explained before, the origin of the track tree.

History: The decision history of the track tree is stored in this field. Decision history gives information about the association decisions of the track in consecutive scans which are an update decision or no-update decision with an observation. This history is used as the consistency measure of the tracks in such a way that a track is consistent if it is updated continuously with the observations in its track gate. The history is stored for the previous n scans in a vector such as $[0 \quad 1 \quad 1 \quad 1]$. This

history vector means that the track is not updated at the $(n-3)^{th}$ scan and it is updated at the next scans. The update case is represented as a 1 and the no update case is represented as a 0 in the vector.

Confirmation flag: A track tree is said to be confirmed if its root nodes are updated for the last *n* consecutive scans. In other words, for the last *n* consecutive scans, the hypotheses used for the decision about the root node are towards updating the root node with observations. History field of the tree is used for confirmation. Confirmation flag is set to 1, when the track tree is confirmed. In this study *n* is 5.

Deletion flag: A track tree is deleted if its deletion flag is 1 and this flag is set to 1 if the root nodes are not updated for the last *n* consecutive scans. In other words, for the last *n* consecutive scans, the hypotheses used for the decision about the root node are towards not updating the root node and the corresponding observations are taken as false alarms. In this study *n* is 5 and the history of a track whose deletion flag is 1 is $[0 \quad 0 \quad 0 \quad 0 \quad 0]$.

Confirmed list number: The confirmed tracks which are the root nodes of the track trees are listed in the confirmed list. And the index of the root node track in the list is stored in this manage track field for the sake of establishing the correspondence between the confirmed list and the manage track structure since the indices of the tracks can be different. Similarly, in the confirmed list the manage track index of the confirmed track is stored.

4.2.7.2 Highest Score Hypothesis Evaluation

In each cluster, the highest score hypothesis of scan *k* is selected for the decision of the associations of scan *k-4*. This evaluation is performed in each cluster since the clusters are independent from each other and the tracks in a cluster are incompatible.

In every cluster, the scores of the hypotheses are ordered in a descending manner. The upper most, highest score, hypothesis is selected if its content is also appropriate. If the highest score hypothesis of scan *k* mentions “the observation is a

false alarm" or "the observation of scan *k* forms a new track", this hypothesis is said to be inappropriate since it does not give an idea to make a decision about scan *k-4*. The aim of the highest score hypothesis selection is to decide which track in the track tree remains alive and which are to be deleted, in other words, which branches are to be cut. Since new tracks are shown as a the initiation of a new track tree whose depth is 1, at scan k, if the highest score hypothesis mentions a new track formation, this evaluation does not give any idea about the update or no-update case of the track at scan *k-4*. Therefore if the highest score hypothesis is a false alarm or a new track initiation the second highest score one is selected and if both the first and second highest score hypotheses are a false alarm or a new track initiation the third highest score hypothesis is selected.

Following this selection, at scan *k-4* the branches whose hypotheses are not in the selected hypothesis are pruned.

4.2.7.3 Tree Pruning

In each cluster, using the highest score hypothesis of scan *k*, tree branches of scan *k-4* are pruned. The highest score hypothesis can mention three cases which are no-association, observation-to-track association and track initiation. Concerning tree structure, no-association corresponds to no-update case of track (track index does not change), observation-to-track association corresponds to observation updates track (a new track index is given to the updated track) and track initiation corresponds to initiation of a new track tree whose root node is this new track.

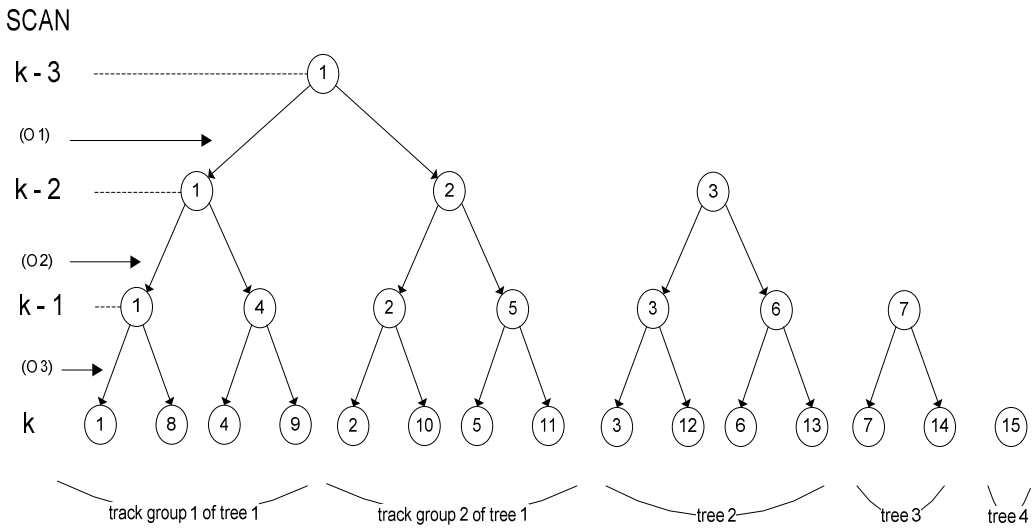


Figure 4-15 Tree pruning example

In Figure 4-15, the decision depth of the track tree is chosen as 3, in this study it is chosen as 5 instead. At scan *k-3*, observation 1 (O1) is in the track gate of track 1 (T1). Therefore, the hypotheses and the tree structure for this scan are as follows:

- No-association (Branch 1): O1 is not associated with T1; T1 is not updated and the index of the track is same; T1 remains as T1 at scan *k-2*.
- Observation-to-track association (Branch 2): O1 updates T1 and forms the updated track T2.
- Track initiation (Branch 3): O1 does not associate with T1; O1 initiates a new track called T3 instead. This new track initiates a new tree and it is the root node of this tree at scan *k-2*.

Highest score hypothesis at scan *k* identifies which branch of scan *k-3* is to be deleted. There are 6 cases that are examined for pruning the branches at scan *k-3* and these are:

- Highest score hypothesis at scan *k* mentions a false alarm: This hypothesis is an inappropriate hypothesis since it mentions that O3 is a false alarm at scan *k*, however it does not give any idea about scan *k-3*. Therefore, in this case

the highest score hypothesis is not used for tree pruning, the next highest score hypothesis is searched until an appropriate one is found.

- Highest score hypothesis at scan *k* belongs to track Group 1 of Tree 1: This group corresponds to Branch 1 of scan *k-3* which means T1 is not updated. Since Branch 1 is selected in scan *k-3*, the other branches (branches whose corresponding hypotheses are opposed to the highest score hypothesis) are deleted as shown in Figure 4-16. Therefore, Branch 2 should be deleted since the update of T1 with O1 is not possible (O1 is not associated with T1). Similarly, Tree 2 is not deleted since initiation of T3 is possible (O1 can initiate a track).

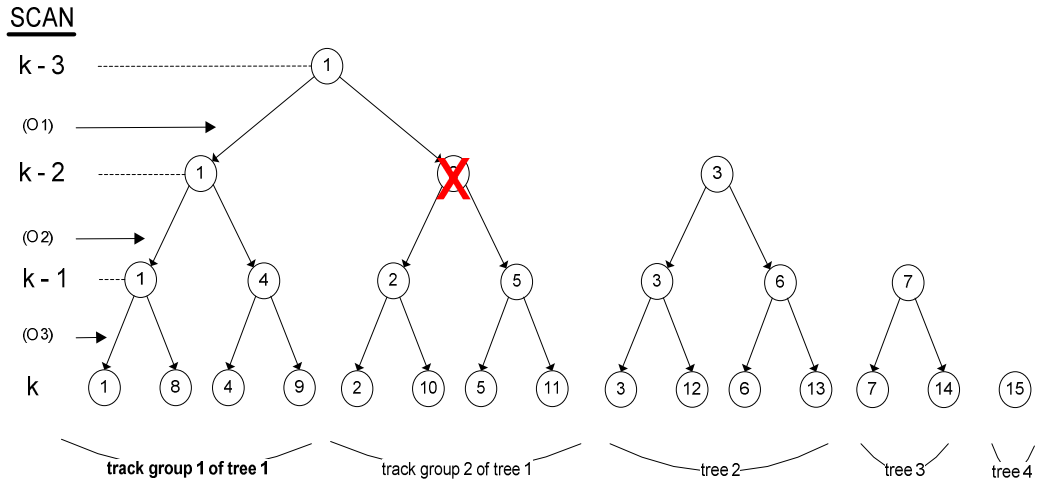


Figure 4-16 Tree pruning example when the highest score hypothesis at scan *k* belongs to track Group 1 of Tree 1

- Highest score hypothesis at scan *k* belongs to track Group 2 of Tree 1: This group corresponds to Branch 2 of scan *k-3* which means O1 updates T1 and forms T2. Since Branch 2 is selected in scan *k-3*, the other branches and newly formed trees of scan *k-3* (branches and trees whose corresponding hypotheses are opposed to the highest score hypothesis) are deleted as shown in Figure 4-17. Therefore, Branch 1 should be deleted since O1

cannot be a false alarm (O1 is updates T1). Similarly, Tree 2 should be deleted since initiation of T3 is not possible (O1 is used for updating T1 and tracks are not allowed to share observations).

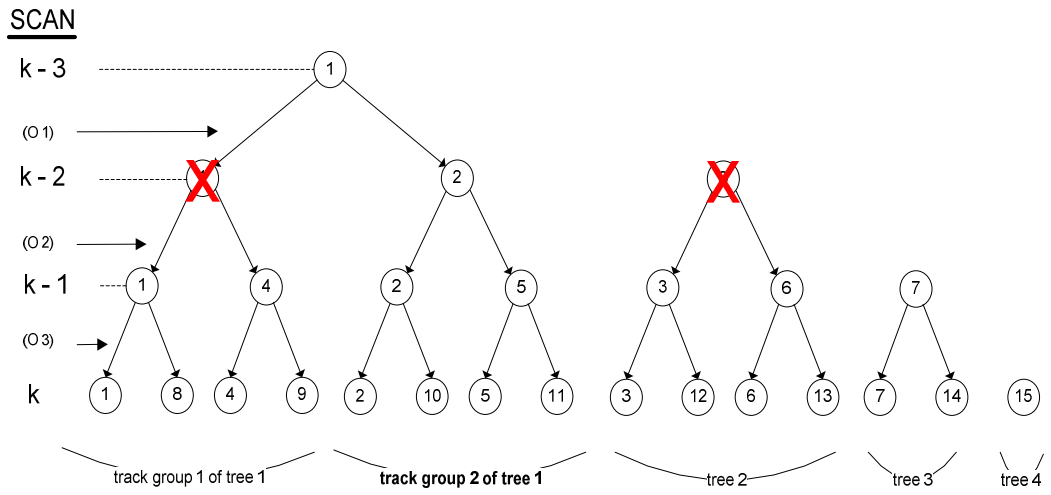


Figure 4-17 Tree pruning example when the highest score hypothesis at scan *k* belongs to track Group 2 of Tree 1

- Highest score hypothesis at scan *k* belongs to Tree 2: Initiation of a new tree of scan *k-3* is selected which means O1 initiates T3. Since Tree 2 is selected in scan *k-3*, the other branches of scan *k-3* (branches whose corresponding hypotheses are opposed to the highest score hypothesis) are deleted as shown in Figure 4-18. Therefore, Branch 2 should be deleted since update of T1 with O1 is not possible (O1 is used for initiating T3 and tracks are not allowed to share observations). Branch 1 should is not deleted since O1 is used to initiate T3 and therefore T1 cannot be associated with O1).

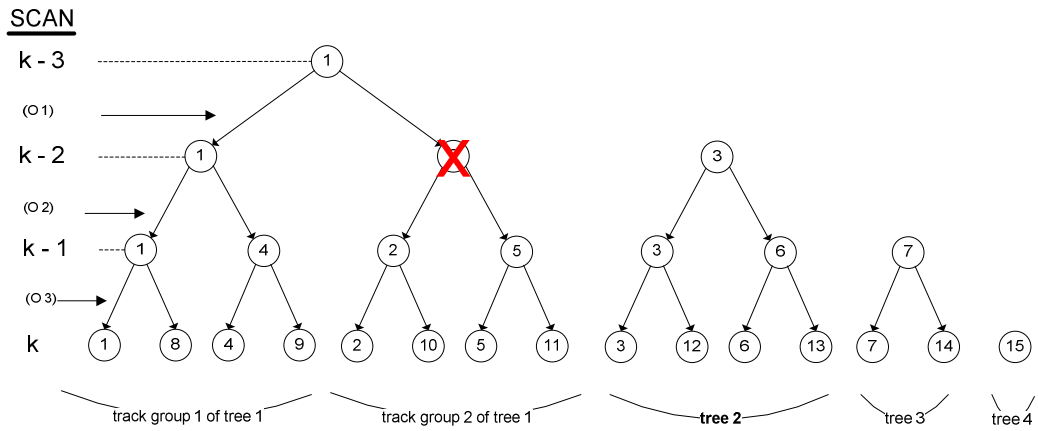


Figure 4-18 Tree pruning example when the highest score hypothesis at scan *k* belongs to Tree 2

- Highest score hypothesis at scan *k* belongs to Tree 3: This hypothesis is an inappropriate hypothesis since it mentions that O3 initiates a new track at scan *k*, however it does not give any idea about scan *k-3*. Therefore, in this case the highest score hypothesis is not used for tree pruning, the next highest score hypothesis is searched until an appropriate one is found.
- Highest score hypothesis at scan *k* belongs to Tree 4: This hypothesis is an inappropriate hypothesis since it mentions that O3 initiates a new track at scan *k*, however it does not give any idea about scan *k-3*. Therefore, in this case the highest score hypothesis is not used for tree pruning, the next highest score hypothesis is searched until an appropriate one is found.

In addition those mentioned up to now, some other pruning operations are performed following these rules:

- If the deletion flag of the track is 1, this track is deleted since this means the track is not updated for the previous 5 scans. In other words, the number of scans during which the track has not been associated with any observation in its gate enough to delete the track.

- If there is a tree whose depth is 5 (more than the allowed depth number in this study) and the highest score hypothesis does not contain any branch of this tree, the branches of the tree that correspond to the update of its tracks are deleted. In other words, its no-update branches remained alive and the tracks of the tree are forced to be deleted in the next scans because of their history (if a track is not updated for consecutive 5 scans, its deletion flag is set to 1 forcing it to be deleted).
- If there is a track tree having only one node and the node is not associated with an observation, this track tree and the track are deleted. This means that the track does not have an observation in its track gate after its initiation and therefore it is not updated and should be deleted.

After pruning operations, current depth, tree number, root node, history, confirmation flag and deletion flag fields of the manage track structure are formed for newly formed tracks and updated for existing tracks. For existing tracks, track indices, track positions, track groups and track scores matrices are updated by 1 column shifting their columns to the left and adding the new values corresponding to the new scan to the right most columns. The updated forms of these matrices are given in Table 4-12, Table 4-13, Table 4-14 and Table 4-15, respectively considering the update case of track 5 in the example tree in Figure 4-14.

Table 4-12 Track indices matrix update for the example track tree in Figure 4-14

Node at depth 1 (root node)	Nodes at depth 2	Nodes at depth 3	Nodes at depth 4
1	1	1	1
0	9	17	40
0	0	9	9
0	0	25	42
0	0	0	17
0	0	0	41
0	0	0	25
0	0	0	43

Table 4-13 Track positions matrix update for the example track tree in Figure 4-14

Node at depth 1 (root node)	Nodes at depth 2	Nodes at depth 3	Nodes at depth 4
2	2	2	0
0	2	2	0
0	0	2	0
0	0	2	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Track groups: The groups of the tracks which are the nodes of the tree are stored in this field. These groups are also held in a matrix called track groups as in Table 4-14. The group number is either 1 or 2 since there are always 2 groups in a tree. The first group is the group of track nodes which are originated from the no update case of the root node, in other words, these tracks are generated from the hypothesis of the root node is not updated with an observation. The second group is the group of track nodes which are originated from the update case of the root node. This notation is necessary for the tree pruning step which will be discussed later since the decision of association is made after k scans. By this way, it is easy to understand which tracks are going to be deleted when the decision is the update or no update of the root node track.

Table 4-14 Track groups matrix update for the example track tree in Figure 4-14

Node at depth 1 (root node)	Nodes at depth 2	Nodes at depth 3	Nodes at depth 4
0	1	1	1
0	2	1	1
0	0	2	2
0	0	2	2
0	0	0	1
0	0	0	1
0	0	0	2
0	0	0	2

Track scores: The scores of the tracks which are the nodes of the tree are stored in this field. These scores are also held in a matrix called track scores as in Table 4-15.

Table 4-15 Track scores matrix update for the example track tree in Figure 4-14

Node at depth 1 (root node)	Nodes at depth 2	Nodes at depth 3	Nodes at depth 4
0.8525	0.8579	0.8605	0.8627
0	0.1421	0.1408	0.5444
0	0	0.6123	0.6144
0	0	0.2842	0.2389
0	0	0	0.1496
0	0	0	0.8253
0	0	0	0.2750
0	0	0	0.1894

4.2.8 Track Confirmation and Occlusion Handling

During a video sequence, many tracks are detected. These tracks can move continuously or stop for a while and then start their motion or stop permanently or disappear for a while, leave the view area of the camera or occlude. In order to be able to track objects accurately, recognize the previously confirmed tracks and handle occlusions some more information about tracks are stored in structures called confirmed track list, occluded track list and occlusion matrix.

Confirmed track list structure is given in Table 4-16 and it is used for establishing the correspondence between confirmed tracks up to that scan and the tracks that are selected from the track trees using the highest score hypothesis.

Table 4-16 Confirmed track list structure for n^{th} track

Confirmed track index
Manage track index
Live
Occluded
Close track index

The explanations of the fields in confirmed track list structure are:

Confirmed track index: This index is given to the tracks that are confirmed. This is the number printed on the screen to mark the tracked objects as the output of the overall system.

Manage track index: This index is the dual of the confirmed list index field in the manage track structure and it is used for establishing the correspondence between confirmed tracks and the tracks in the track trees.

Live: It is the flag that shows a confirmed track is whether alive or dead. Here, an alive track means the tracks is not deleted in its corresponding track tree and a dead track means vice versa. Dead tracks are the tracks that leave the view area of the camera or stop for a while or permanently, or pass behind an object.

Occluded: It is the flag that shows two or more objects get too close to each other that they are detected as one track. Occlusion is the detection and tracking of more than one object as one track because of their closeness. This flag is set to 1 if the confirmed track is occluded with other track/tracks.

Close track index: The indices of the tracks that are close to that confirmed track are stored in this field. This information is used for occlusion detection.

4.2.8.1 Confirmed Track Management

It is very important to be consistent on naming confirmed tracks while managing them. The tracks are named with their confirmed track list indices and these indices are printed on the output video to mark the tracks.

New confirmed track list indices are given to new tracks. Each time a new confirmed track is detected, it is compared with the deleted (dead) tracks since dead tracks may appear in the following frames. This can happen when an object stops for a while and then moves again or when an object passes behind an object, etc. Since moving objects are detected, when an object stops it is not detected and the corresponding track dies, similarly when a track moves behind an object it is not detected and the corresponding track dies. However, after a while when the object is detected again, it is tracked with a new track tree index. Therefore, to establish the correspondence between track trees and printed track indices a confirmed track list is used in this study. The track index in the manage track structure is stored in the manage track number field of the confirmed track list and vice versa.

The algorithm for confirmed track management is given in Figure 4-19.

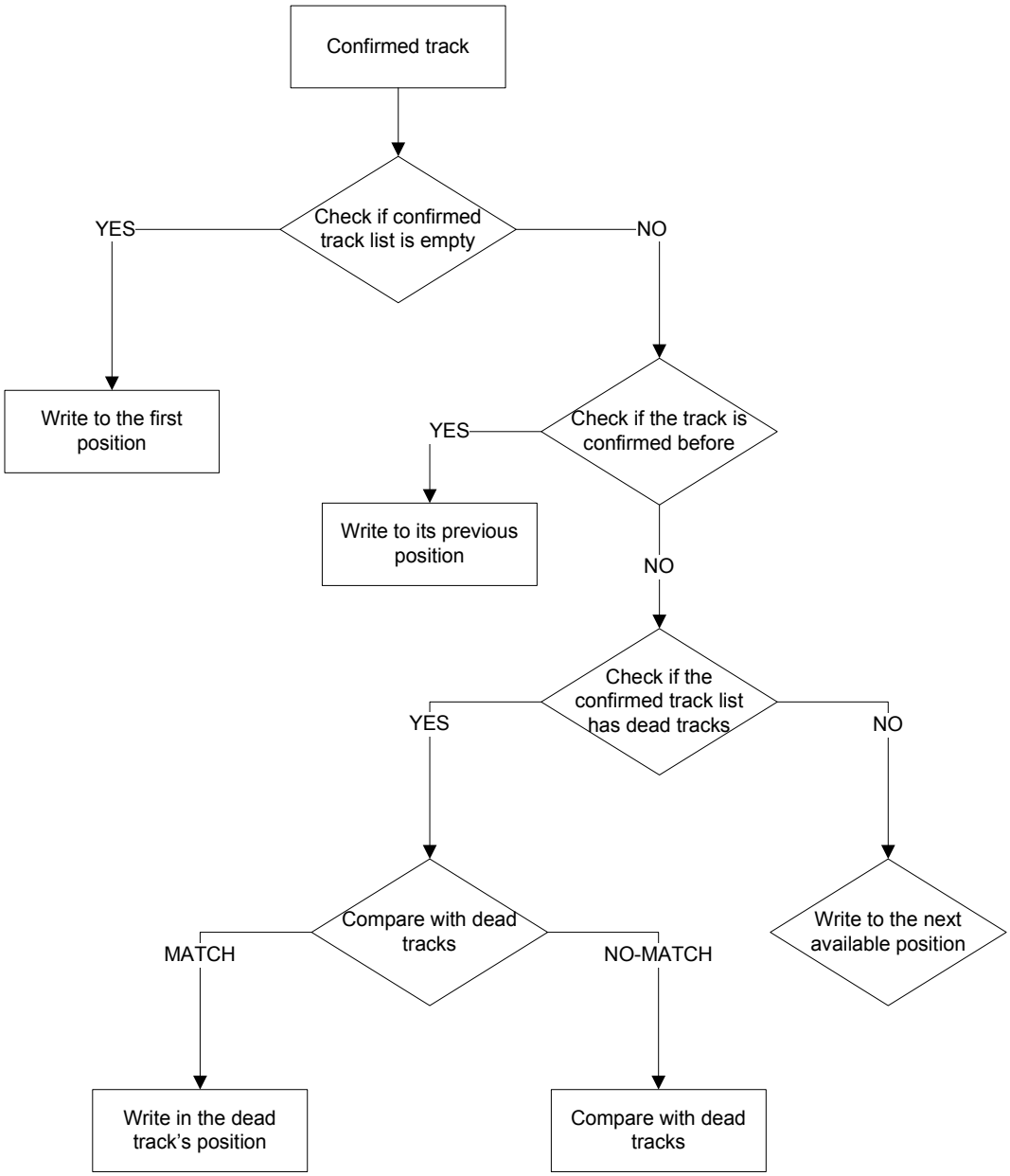


Figure 4-19 Confirmed track management algorithm

The algorithm can be summarized as follows:

- If the confirmed track list is empty, the confirmed track is written to the first position in the list.

- If the confirmed track list is not empty and the confirmed track has already been in the confirmed track list with that track number before, the confirmed track is written to its previous position in the list.
- If the confirmed track list is not empty and the confirmed track has never been in the confirmed track list with that track number before and there is no dead tracks in the confirmed track list, the confirmed track is written to the next available position in the list.
- If the confirmed track list is not empty and the confirmed track has never been in the confirmed track list with that track number before and there are dead tracks in the confirmed track list, a comparison is made to decide whether the confirmed track is a previously dead track. This comparison is made concerning position, size, and color of the tracks. A match is established when

$$\sqrt{(x_l - x_d)^2 + (y_l - y_d)^2} \leq T_{pos} \tag{4-19}$$

and

$$\left|\mu_{RGB_l} - \mu_{RGB_d}\right| \leq T_{col} \tag{4-20}$$

and

$$\left|s_l - s_d\right| \leq T_{size} \tag{4-21}$$

where T_{pos} is the position threshold, T_{col} is the RGB color threshold, T_{size} is the size threshold, $c_l(x_l, y_l)$ is the center of the live track, $c_d(x_d, y_d)$ is the center of the dead track, μ_{RGB_l} is the RGB color mean of the live (confirmed) track, μ_{RGB_d} is the RGB color mean of the dead track, s_l is the size of the live track and s_d is the size of the dead track. μ_{RGB} of live and dead tracks are calculated as follows:

$$\mu_{RGB} = \frac{\mu_R + \mu_G + \mu_B}{3} \tag{4-22}$$

If there is a match between the confirmed track and the dead track, the confirmed track is written in the dead track's position.

- If the confirmed track list is not empty and the confirmed track has never been in the confirmed track list with that track number before and there are dead tracks in the confirmed track list, a comparison is made to decide whether the confirmed track is a previously dead track. If there is no match between the confirmed track and the dead tracks, the confirmed track is written to the next available position in the list.

4.2.8.2 Occlusion Handling

Occlusion handling is performed in two steps: Occlusion Detection and Separation of Occlusion Participants as shown in Figure 4-20.

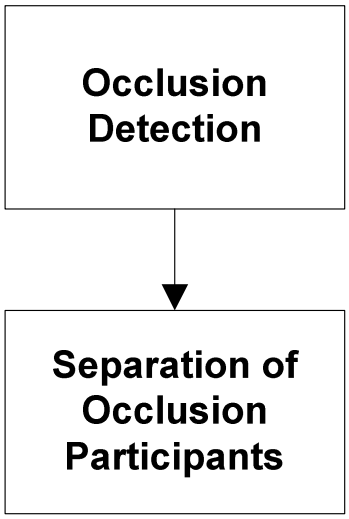


Figure 4-20 Flow diagram of occlusion handling

Occlusions are handled using occluded track list and occlusion matrix in addition to confirmed track list. Occluded track list structure is given in Table 4-17.

Table 4-17 Occluded track list structure for n[th] track

Occlusion participants
Occlusion center
Occlusion radius
Resultant occluded track

The explanations of the fields in the occluded track list structure are:

Occlusion participants: The indices of the occlusion participants are stored in this field. The number of participant tracks can be more than two.

Occlusion center: The arithmetic mean of the x-y coordinates of the participant tracks is stored in this field. The track center is the centroid of the area that approximates the region of the track. Occlusion center (x_{oc}, y_{oc}) is calculated as follows:

$$(x_{oc}, y_{oc}) = \left(\frac{\sum_{i=0}^{n} x_{ic}}{n}, \frac{\sum_{i=0}^{n} y_{ic}}{n} \right) \tag{4-23}$$

where (x_{ic}, y_{ic}) is the center of the i^{th} track in the occlusion.

Occlusion radius: The sum of the radii of the participant tracks is stored in this field. The track radius is the radius of the circle that approximates the region of the track. Occlusion radius r_o is calculated as follows:

$$r_o = \sum_{i=0}^{n} r_i \tag{4-24}$$

where r_i is the radius of the i^{th} track in the occlusion.

Resultant occluded track: The track index which is formed as the result of the occlusion and whose center is (x_{ic}, y_{ic}) and radius is r_o.

Table 4-18 Occlusion matrix for n confirmed tracks where $n = 4$

	Track1	Track2	Track3	Track4
Track1	0	d_{12}	d_{13}	d_{14}
Track2	d_{21}	0	d_{23}	d_{24}
Track3	d_{31}	d_{32}	0	d_{34}
Track4	d_{41}	d_{42}	d_{43}	0

Occlusion matrix, which is shown in Table 4-18, is formed to detect occlusions using the distances between live confirmed tracks. The confirmed tracks are said to be occluded if the Euclidean distance between track centers is smaller than the sum of the radii of the tracks plus a threshold and a new track appears in the thresholded area where the confirmed close tracks disappeared.

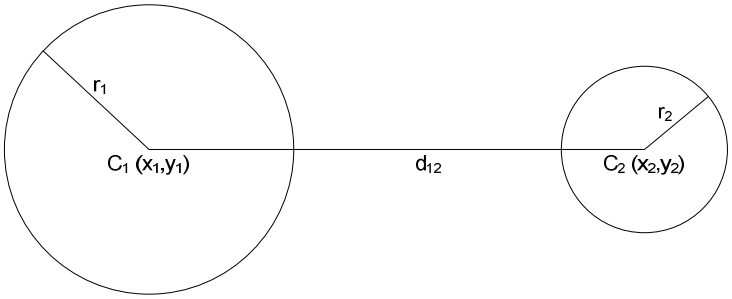


Figure 4-21 Occlusion of tracks

In the occlusion matrix, the Euclidean distances between each track is stored in its cells. Occlusion of two tracks is shown in Figure 4-21 where r_1, r_2 are the radii and

$c_1(x_1, y_1)$ and $c_2(x_2, y_2)$ are the centers of the tracks T1 and T2, respectively. The Euclidean distance d_{12} is calculated as follows for the tracks:

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4-25}$$

The tracks are said to be occluded if

$$d_{12} - (r_1 + r_2) \leq T_o \tag{4-26}$$

where T_o is the occlusion threshold used as a measure of the closeness of tracks. In the occlusion matrix, $d_{ij} = d_{ji}$ since they refer the same distance and $d_{ii} = 0$ since the diagonals in the matrix correspond to the distances of the tracks to themselves.

4.2.8.2.1 Occlusion Detection

Occlusion occurs when two or more tracks get closer to each other so that they are detected as one object. The resultant object is tracked as a new track in this study. And this new track, resultant occluded track, has a new track index. The flow of the occlusion detection algorithm is given in Figure 4-22.

Find an occlusion participant: The occlusion participants are not-occluded dead tracks in that scan. Using the confirmed track list, the tracks whose occlusion and live fields are 0 are found.

Find the other occlusion participant/participants: From the occlusion matrix, find other not-occluded dead tracks that are close to tracks found in the previous step.

Update the occlusion track list: Add the occlusion participants to each others' occlusion track list. Calculate the occlusion center and occlusion radius fields of the occlusion list.

Find the resultant occluded track: Check whether a new track is formed in the occlusion region. Here, the occlusion region is the circle centered at the occlusion center with a radius of the occlusion radius. This newly formed track is the resultant track of the occlusion, in other words, this track is formed since two tracks are too

close to each other that they are detected as one track. This track is stored in the resultant occluded track field of the occlude track list.

Update confirmed track list: An occlusion is detected therefore the occluded flag in the confirmation track list of the dead tracks that participate in the occlusion are set to 1.

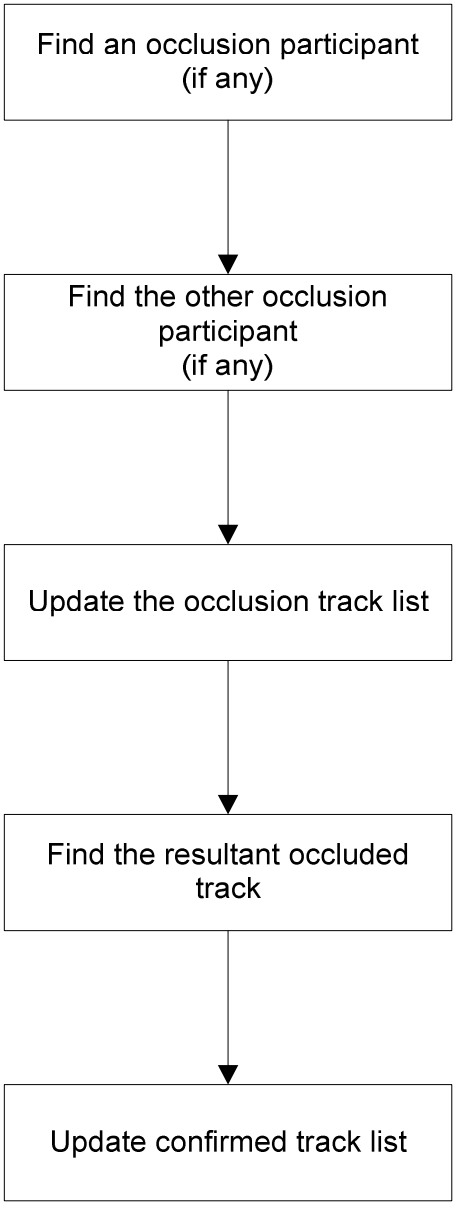


Figure 4-22 Occlusion detection algorithm

4.2.8.2.2 Separation of Occlusion Participants:

Occlusion of objects ends when the occlusion participants separate from each other. The main problem in the end of the occlusion is the matching of the occlusion participants with the new tracks that are initiated in the region where the occlusion ends. The flow of the separation of the occlusion participants algorithm is given in Figure 4-23.

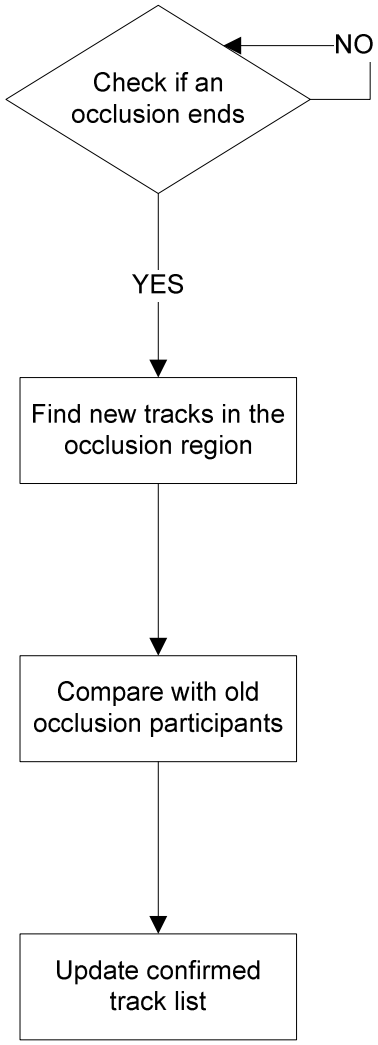


Figure 4-23 Separation of the occlusion participants algorithm

Check if an occlusion ends: End of an occlusion can be understood from the live flag (in the confirmed track list) of the new formed track (resultant occluded track)

in the occlude track list. If this flag is set to 0, the occlusion which results in the formation of that new formed track ends.

Find new tracks in the occlusion region: Using the confirmed track list, examine the new tracks whether they are initiated in the region where the occlusion ends. The examination is performed by calculating the distance between the centers of the occluded track $c_o(x_o, y_o)$ and the new tracks $c_n(x_n, y_n)$ that are detected in the region of occlusion and comparing this distance with a distance threshold. In this study this distance threshold is chosen as 2 times the radius of the occluded track, r_o:

$$\sqrt{(x_n - x_o)^2 + (y_n - y_o)^2} \leq 2 * r_o \tag{4-27}$$

If this condition is satisfied then, the new tracks are said to be in the region where the occlusion ends and they may be formed because of the separation of the occluded tracks.

Compare with old occlusion participants: In order to match the old occlusion participants with the new tracks that are formed at the end of an occlusion, a comparison is made by calculating the separation score. This score, $score_{sep}$, is evaluated according to the size and color of the tracks as follows:

$$score_{sep} = w_c * \left| \mu_{RGB_{new}} - \mu_{RGB_{old}} \right| + w_s * \left| s_{new} - s_{old} \right| \tag{4-28}$$

where w_c is the RGB color weight, w_s is the size weight, $\mu_{RGB_{new}}$ is the RGB color mean of the new track, $\mu_{RGB_{old}}$ is the RGB color mean of the old track, s_{new} is the size of the new track and s_{old} is the size of the old track. Here an old track is one of the occlusion participants (from the occlusion detection algorithm) and occlusion participants are previously confirmed dead tracks whose occluded flag is 1 in the confirmed track list. μ_{RGB} of both new and old tracks are calculated as follows:

$$\mu_{RGB} = \frac{\mu_R + \mu_G + \mu_B}{3} \tag{4-29}$$

The separation scores, $score_{sep}$, corresponding to occlusion participants and the new tracks are calculated and sorted. The lowest score is compared with the separation threshold, T_{sep}, if the lowest separation score is also lower than the separation threshold, then a match is found. The new track is matched with the one that is dead because of a previous occlusion.

Update confirmed list: In the confirmed track list, the track indices of the new confirmed tracks that are initiated in the region of occlusion are replaced with the matched track indices of the occlusion participants.

4.3 Results

In this section, PETS2001/Dataset1/Testing/Camera1 video [30] is used as "test video1", PETS2001/Dataset1/Training/Camera1 video [30] is used as "test video2", PETS2001/Dataset2/Testing/Camera2 video [30] is used as "test video3" and PETS2006/S4-T5-A video [31] is used as "test video4".

4.3.1 Results of the Developed System with Multiple Targets

In Figure 4-24 and Figure 4-25, some examples of the performance of the overall system are presented for tracking multiple targets case.

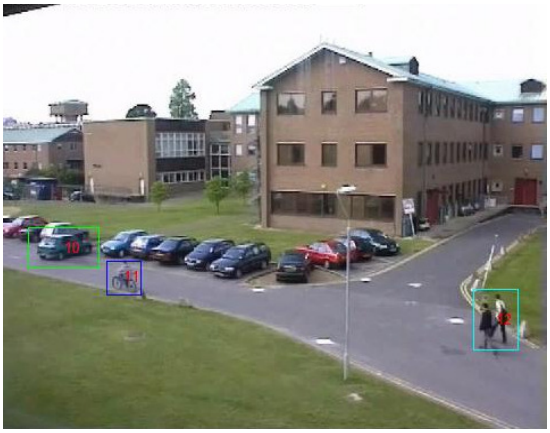


Figure 4-24 An example from test video3 – Tracking of multiple targets labeled as 10, 11 and 12

Figure 4-25 An example from test video4 – Tracking of multiple targets labeled as 1, 2 and 3

4.3.2 Results of the Developed System in Occlusion Case

In this section, some examples of the occlusion handling results of the overall system are presented through Figure 4-26 to Figure 4-34. In order to show the results of each occlusion case, three scenes of the test videos are selected; one scene to represent the participants before occlusion, one scene to represent the resultant occluded track and one scene to represent the participants after occlusion.

Figure 4-26 An occlusion example from test video1 – Before occlusion of tracks labeled 5 and 6

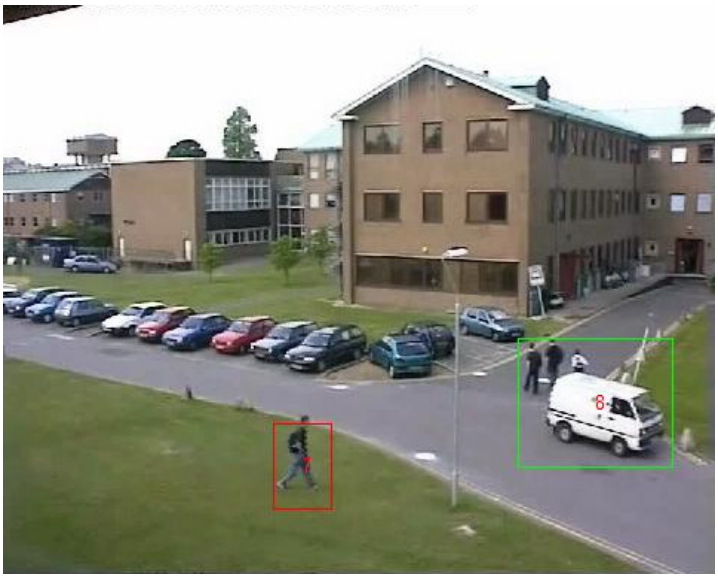


Figure 4-27 An occlusion example from test video1 – During occlusion, the occlusion of tracks labeled as 5 and 6, results in the occluded track labeled as 8

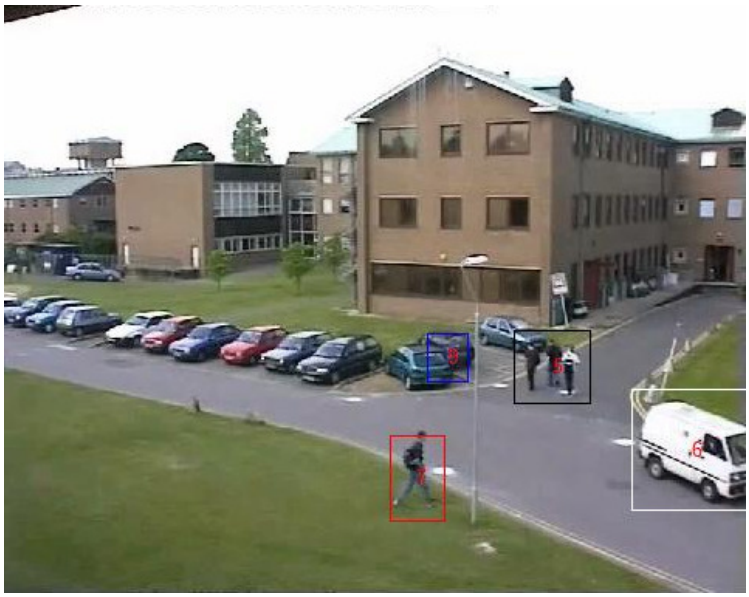


Figure 4-28 An occlusion example from test video1 – At the end of the occlusion the occlusion participants labeled as track 5 and track 6 are separated

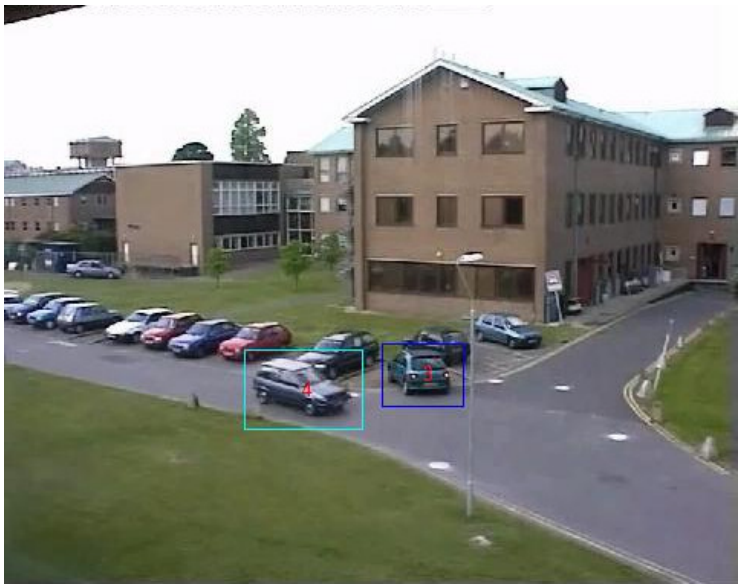


Figure 4-29 An occlusion example from test video2 – Before occlusion of tracks labeled 3 and 4

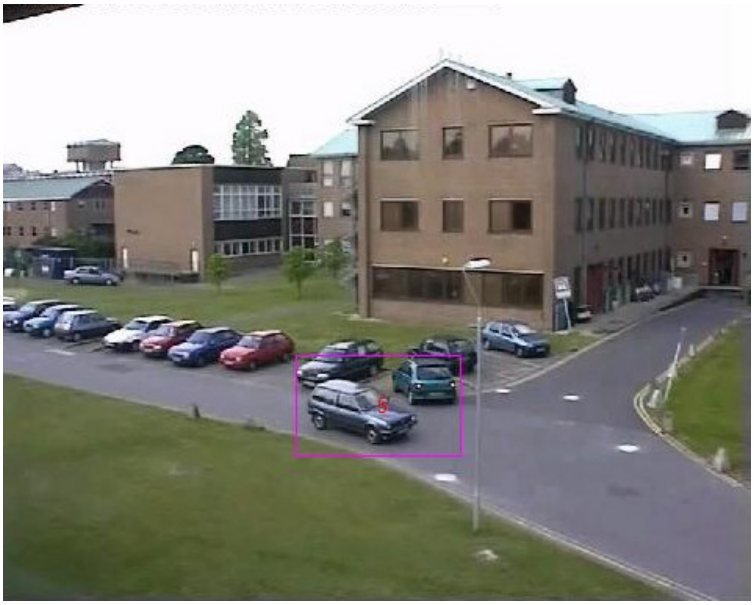


Figure 4-30 An occlusion example from test video2 – During occlusion, the occlusion of tracks labeled as 3 and 4, results in the occluded track labeled as 5

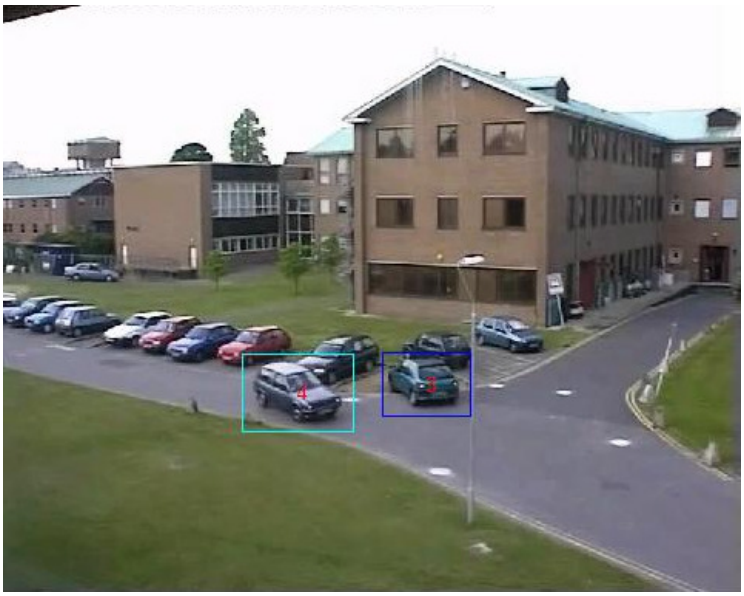


Figure 4-31 An occlusion example from test video2 – At the end of the occlusion the occlusion participants labeled as track 3 and track 4 are separated

Figure 4-32 An occlusion example from test video4– Before occlusion of tracks labeled 8 and 9

Figure 4-33 An occlusion example from test video4 – During occlusion, the occlusion of tracks labeled as 8 and 9, results in the occluded track labeled as 10

Figure 4-34 An occlusion example from test video4 – At the end of the occlusion the occlusion participants labeled as track 8 and track 9 are separated

CHAPTER 5

PERFORMANCE ANALYSIS OF MULTIPLE HYPOTHESIS TRACKING ALGORITHM

Multiple Hypothesis Tracking algorithm is implemented and tested over some scenarios using PETS2001 [30] and PETS2006 [31] video sets. Some of the videos from these video sets are chosen to test and examine the sensitivity of the MHT algorithm according to their scenarios in this study. Since the run time of the implemented code is long, some periods of these selected videos are used. In this chapter background training, target tracking and occlusion handling performances of the MHT implementation is analyzed under some conditions like indoor and outdoor videos, tracking close and distant targets.

5.1 Tracking Performance Analysis According to the Background Training Duration

This analysis aims to examine the sensitivity of the performance of the tracker to the number of frames that are used for background training in foreground segmentation step. This analysis is performed on PETS2006/S4-T5-A video [31].

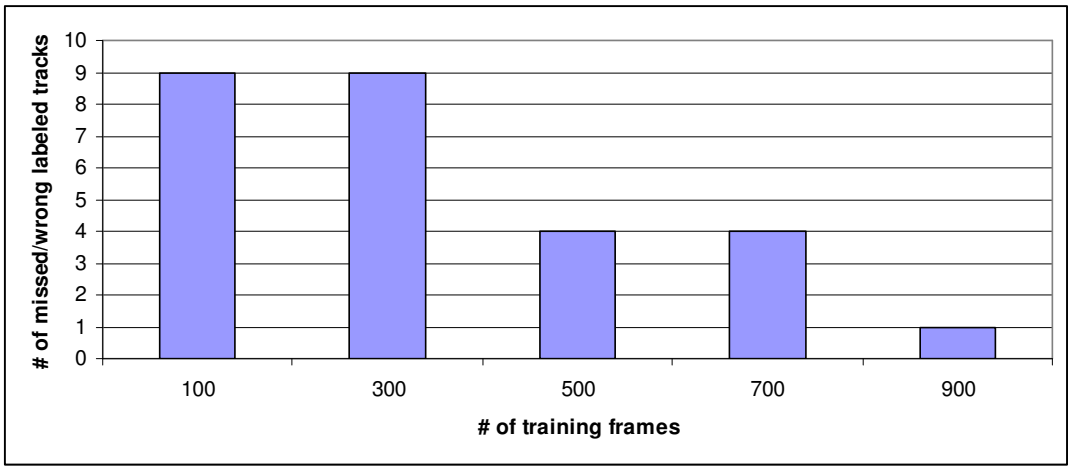


Figure 5-1 Sensitivity of the tracker to the background training frame number

The performance of the overall tracker is highly dependent on the performance of the foreground segmentation step. As explained in section 3.1, Mixture of Gaussians method is used for moving object detection in this study. Background training period is effective on the detection performance and as shown in Figure 5-1, the performance of the tracker increases with the number of background training frames. This is because, using a relatively small learning rate (i.e., 0.0008) and a long learning period (i.e., 900 frames) results in a well-defined background, successful moving object detection and therefore less mistakes in moving object tracking. In this study, before tracking algorithm is run, background training is performed with a small background learning rate and a duration of nearly the same number as the frame number of the test video. After this training period, the overall algorithm is run and background learning rate is increased (i.e., 0.008) so that the foreground segmentation algorithm would be able to include the objects, which stop after a while, in the background in a fast manner.

5.2 Tracking Performance Analysis According to the Parameters Used in Hypothesis Score Calculation

This analysis aims to examine the sensitivity of the tracking performance of MHT algorithm to the parameters used in hypothesis score calculation. In other words, the

aim is to understand how observation-to-track association is affected by the features such as distance, size, color, gate size and track history.

In this analysis PETS2001/Dataset1/Testing/Camera1 video is used as “test video1”, PETS2001/Dataset1/Training/Camera1 video is used as “test video2” and PETS2006/S4-T5-A video is used as “test video3”. The features of the input test videos are given in Table 5-1.

Table 5-1 Features of the input test videos

Input/Features	Environment	Distance of the camera	Number of moving objects	Illumination
Test video1	Outdoor	Distant	6	Daylight
Test video2	Outdoor	Distant	8	Daylight
Test video3	Indoor	Close	7	Indoor lightning

Test video1 and test video2 contain both close and distant objects whereas test video3 contains close objects. Here, close and distant words are used for defining the distance between targets and the camera. As mentioned in section 4.2.4, distance, size, color, gate size and track history features are used in hypothesis score calculation with weights changing from 0 to 1 in the score formula (4-3). The feature weights that are used in hypothesis score calculation of tracking performance analysis for test videos are given in Table 5-2.

Table 5-2 Feature weights used in hypothesis score calculation of tracking performance analysis for test videos

Input/Parameter	Distance	Size	Color	Gate	History
Test video1	1	0.2	0.1	0.4	0.8
Test video2	1	0.2	0.1	0.6	0.8
Test video3	0.8	0.2	0.1	0.2	0.8

The sensitivity of the MHT algorithm is examined according to one parameter by giving different values to that parameter while keeping the other parameters same as in Table 5-2. The tracking performance analysis result is given in Table 5-3.

Table 5-3 Sensitivity of the tracking performance of MHT algorithm in terms of the parameters used in hypothesis score calculation

Condition/Parameter	Distance	Size	Color	Gate	History
Indoor	insensitive	insensitive	insensitive	sensitive	sensitive
Outdoor	insensitive	insensitive	sensitive	sensitive	sensitive
Close Objects	sensitive	sensitive	insensitive	sensitive	sensitive
Distant Objects	insensitive	insensitive	insensitive	sensitive	sensitive

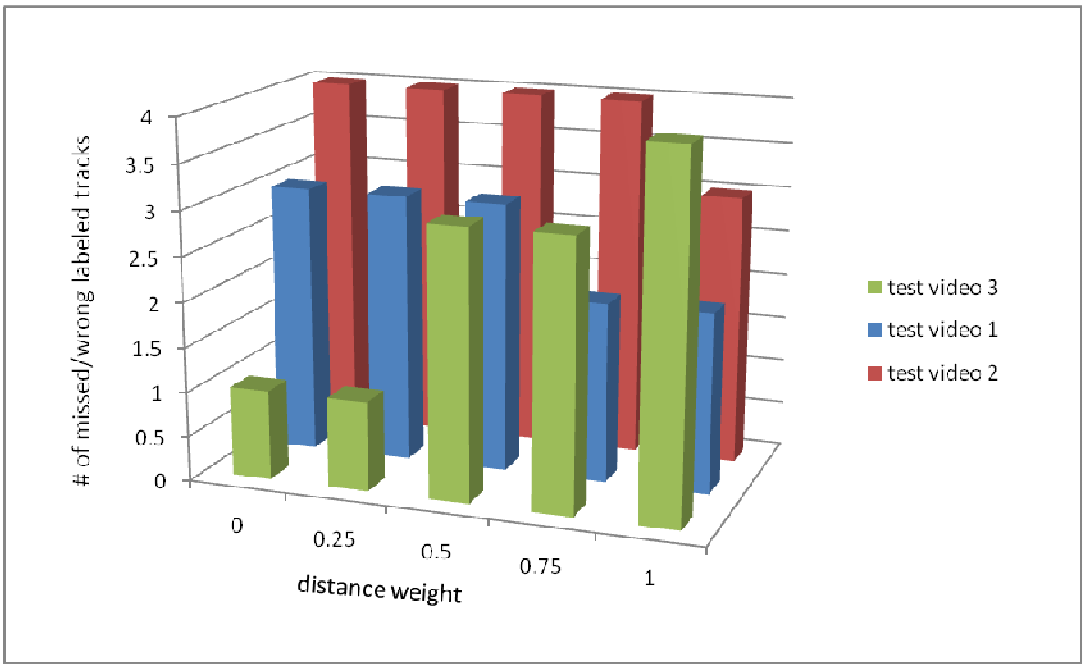


Figure 5-2 Sensitivity graph of the tracking performance of MHT algorithm in terms of distance weight in hypothesis score calculation

According to the analysis results given in Figure 5-2, the algorithm is not sensitive to the changes in the weight of the distance feature while tracking distant objects (objects far from the camera) whereas it is sensitive in close objects (objects close to the camera) case. The reason is that in consecutive frames the displacement of a distant object is small; therefore the observations in the gate of a distant track are very close to that track and observation-to-track associations are succeeded. However, the displacement of a close object in consecutive frames is greater than that of a distant one and the observations in the gate of a close track are not close to that track. Therefore, while tracking close objects hypothesis score is sensitive to distance feature and observation-to-track associations are not always succeeded.

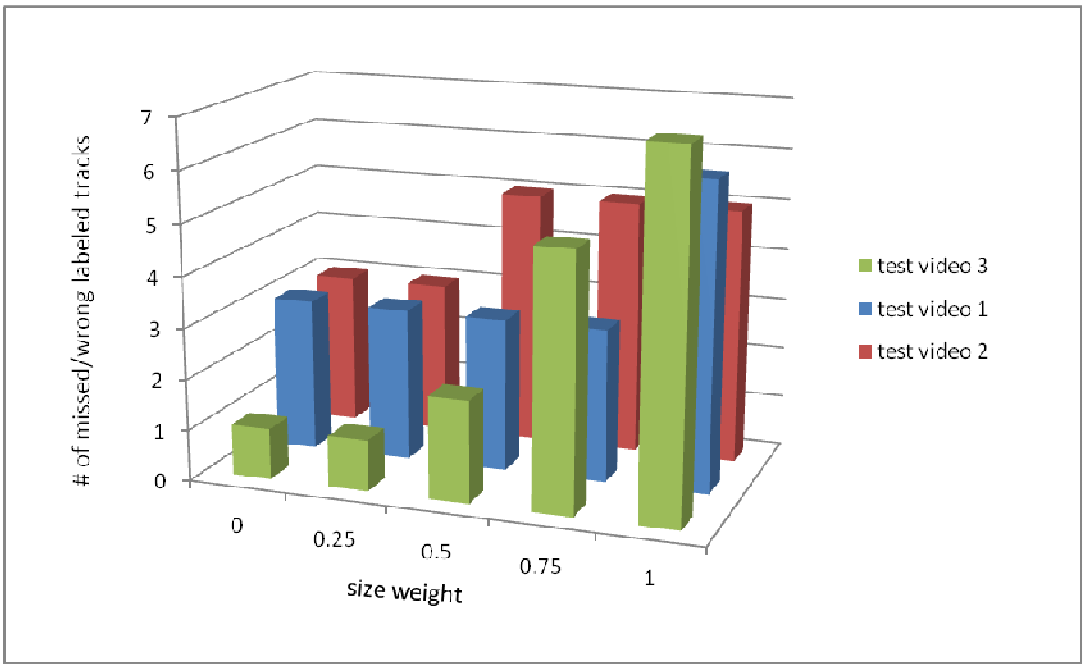


Figure 5-3 Sensitivity graph of the tracking performance of MHT algorithm in terms of size weight in hypothesis score calculation

The algorithm is not sensitive to the changes in the weight of the size feature while tracking distant objects whereas it is sensitive in close objects case as shown in Figure 5-3. Since a distant object is small with respect to a close one, in consecutive frames the size difference of a distant object is very small. Therefore the sizes of the observations in the gate of a distant track are very similar to the size of that track and consequently in hypothesis score calculation the size score is generally high and the algorithm is not sensitive to size feature for distant tracks.

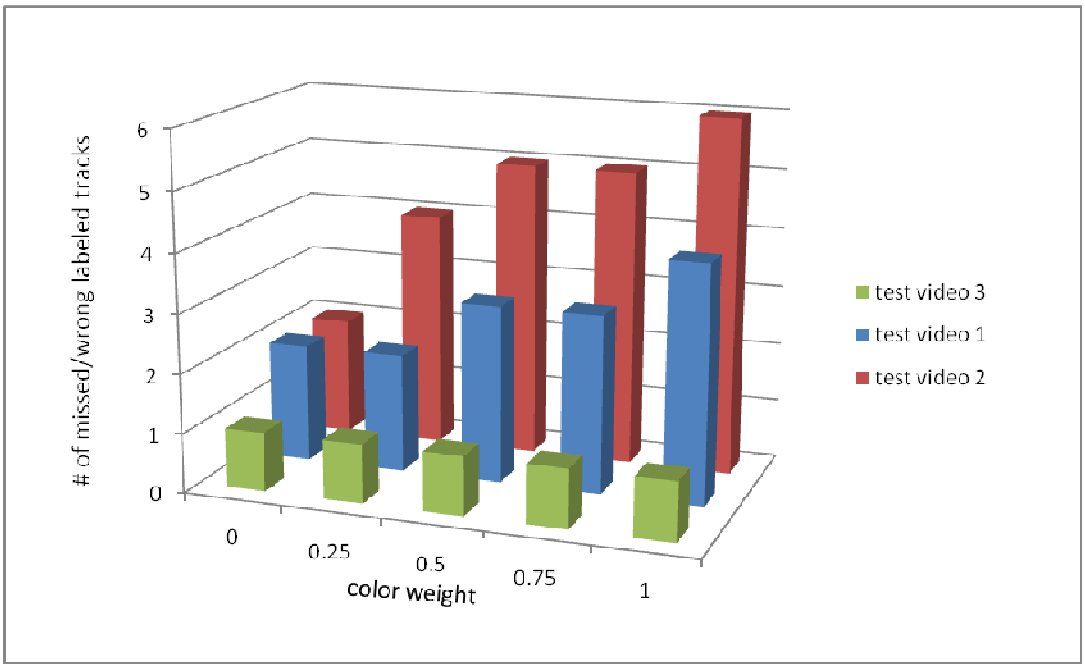


Figure 5-4 Sensitivity graph of the tracking performance of MHT algorithm in terms of color weight in hypothesis score calculation

Another result of the analysis is that the algorithm is not sensitive to the changes in the weight of the color feature when tested with indoor videos whereas it is sensitive when tested with outdoor videos as shown in Figure 5-4. This is because in consecutive frames of indoor videos, lightning is almost stable however; in outdoor conditions lightning may change abruptly and therefore the colors of the detected objects may differ from frame to frame as a result of too much lightning or shadows. Consequently, for outdoor scenes the color of the observations in the gate of a track may be different from the color of that track and therefore in hypothesis score calculation the color score is generally low and the algorithm is sensitive to color feature for outdoor videos.

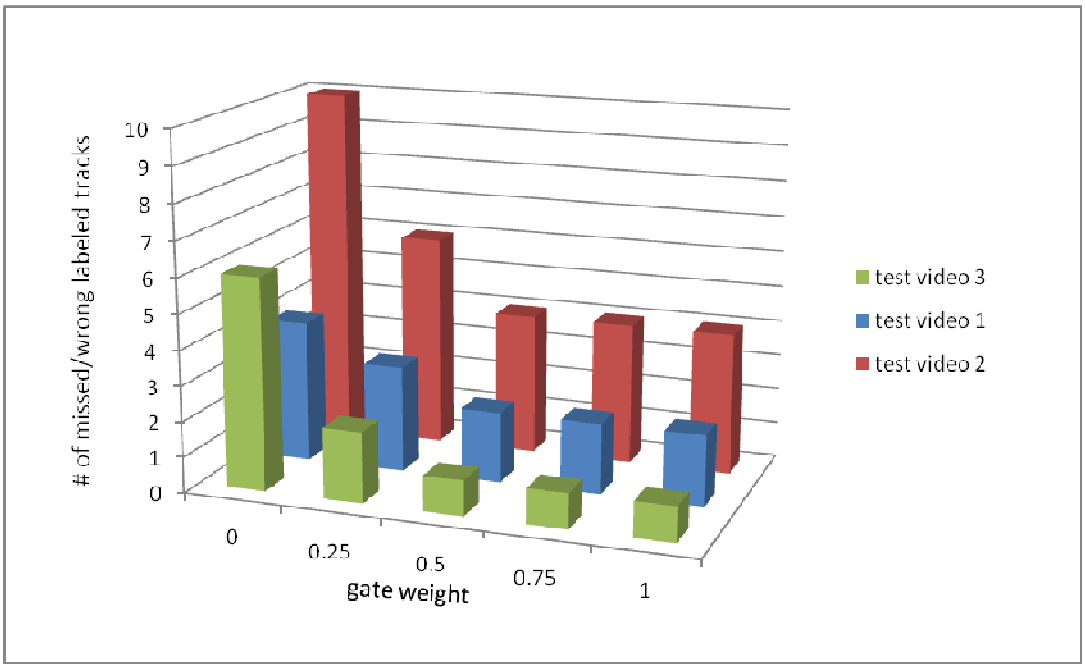


Figure 5-5 Sensitivity graph of the tracking performance of MHT algorithm in terms of gate size weight in hypothesis score calculation

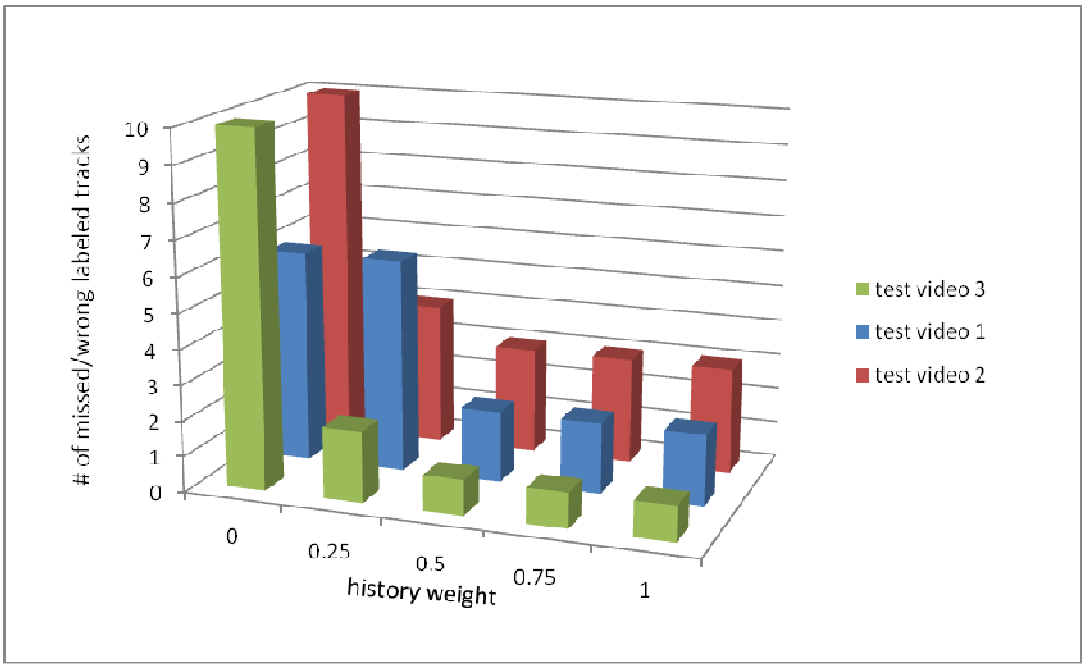


Figure 5-6 Sensitivity graph of the tracking performance of MHT algorithm in terms of history weight in hypothesis score calculation

The analysis results also imply that the algorithm is sensitive to the changes in the gate size and history weights in any condition as shown in Figure 5-5 and Figure 5-6. In fact, the sensitivity of the algorithm to these features is slightly different since these features are the consistency measures of the tracks. In hypothesis score calculation, gate and history scores give idea about the previous associations of a track. In detail, if the track is continuously updated (associated) with the observations in its track gate, its gate becomes smaller and therefore its gate score becomes higher. Similarly, if the track is continuously updated (associated) with the observations in its track gate, the history score of the track becomes higher since the update of the tracks with the observations empowers the consistency of tracking. Therefore, for all conditions of the analysis, with a weight value above 0.25 the performance of the algorithm is high therefore; these parameters may be fixed for all conditions above 0.25.

According to the tracking performance analysis the optimum feature weights are determined for each test video as in Table 5-4.

Table 5-4 Optimum feature weights used in hypothesis score calculation of tracking performance analysis for test videos

Input/Parameter	Distance	Size	Color	Gate	History
Test video1	1	0	0	0.75	0.75
Test video2	1	0	0	0.75	0.75
Test video3	0	0	0.5	0.75	0.75

5.3 Occlusion Handling Performance Analysis According to the Parameters Used in Separation Score Calculation

This analysis aims to examine the sensitivity of the occlusion handling performance of the MHT algorithm to the parameters used in separation score calculation. In other words, the aim is to understand how the separation of occlusion participants' performance is affected by the features such as size and color.

In this analysis PETS2001/Dataset1/Testing/Camera1 video is used as "test video1", PETS2001/Dataset1/Training/Camera1 video is used as "test video2" and PETS2006/S4-T5-A video is used as "test video3". The features of the input test videos are given in Table 5-5.

Table 5-5 Features of the input test videos

Input/Features	Environment	Distance of the camera	Illumination	Number of occlusions
Test video1	Outdoor	Distant	Daylight	1
Test video2	Outdoor	Distant	Daylight	3
Test video3	Indoor	Close	Indoor lightning	2

Test video1 and test video2 contain both close and distant objects whereas test video3 contains close objects. Here, close and distant words are used for defining the distance between targets and the camera. As mentioned in section 4.2.8.2.2, at the end of an occlusion, occlusion participants are compared with the new tracks that are initiated in the region where the occlusion ends. This comparison is made considering the separation score. Size and color features are used in separation

calculation with weights changing from 0 to 1 in the separation score formula (4-28). The feature weights that are used in separation score calculation for test videos are given in Table 5-6.

Table 5-6 Feature weights used in separation score calculation for test videos

Input/Parameter	Size	Color
Test video1	0	0.5
Test video2	0	0.5
Test video3	0	0.5

The sensitivity of the MHT algorithm is examined according to one parameter by giving different values to that parameter while keeping the other parameter same as in Table 5-6. The tracking performance analysis result is given in Table 5-7.

Table 5-7 Sensitivity of the occlusion handling performance of the MHT algorithm in terms of the parameters used in separation score calculation

Condition/Parameter	Size	Color
Indoor	insensitive	sensitive
Outdoor	insensitive	sensitive
Close Objects	insensitive	sensitive
Distant Objects	insensitive	sensitive

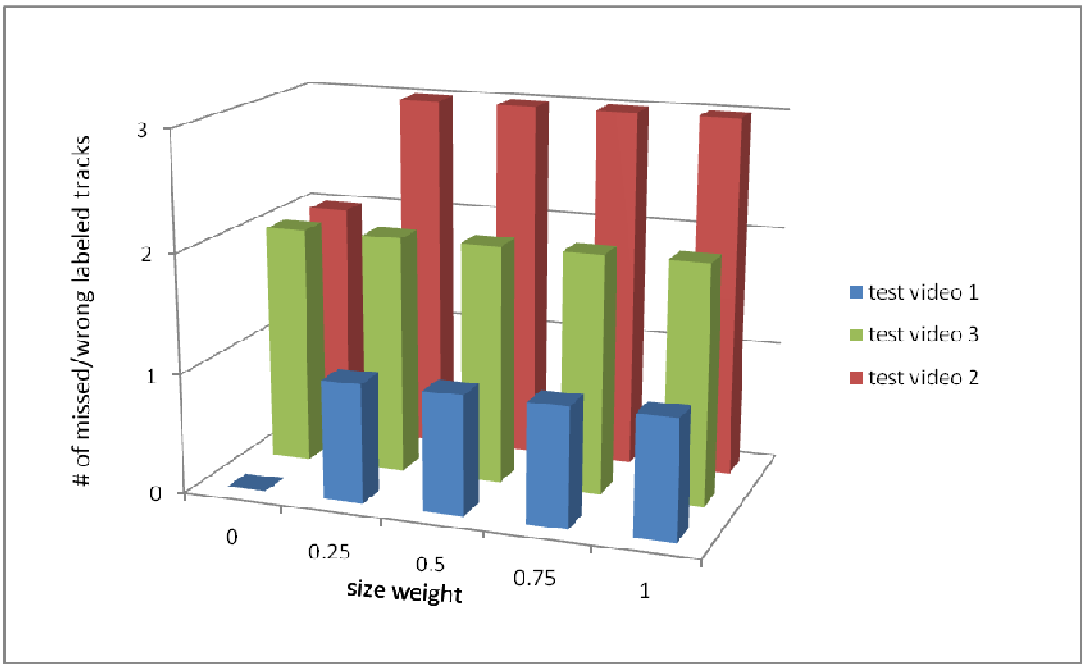


Figure 5-7 Sensitivity graph of the occlusion handling performance of MHT algorithm in terms of size weight in separation score calculation

According to the analysis as shown in Figure 5-7, the algorithm is insensitive to the changes in the weight of the size feature for all cases. The increase in the weight of the size feature results in a decrease in the performance of the system. This is because during the occlusion, the sizes of the participants change as they move towards the camera or far from the camera. As a result, before and after the occlusion, the participants' sizes are different and it is not correct to match the participants considering their sizes for these test videos. Therefore, size is not a distinctive feature between the occlusion participants for the test videos that are used in this analysis. For some other test scenarios that have occlusion participants with different sizes, a size weight greater than 0 would help in matching the participants.

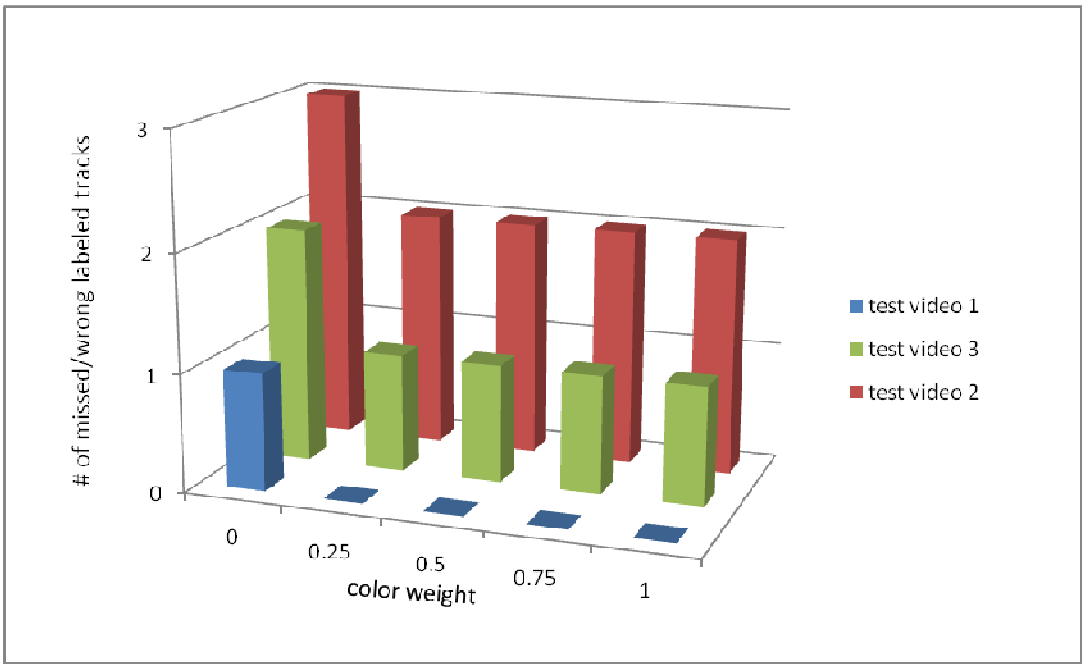


Figure 5-8 Sensitivity graph of the occlusion handling performance of MHT algorithm in terms of color weight in separation score calculation

The algorithm is sensitive to the changes in the weight of the color feature for all conditions as shown in Figure 5-8. In fact, the sensitivity of the algorithm to this feature is slightly different; because for these test videos, using color feature is a necessity for occlusion handling and for color weights above 0.25 the performance of the algorithm does not change. During the occlusion, the colors of the participants do not change much and as a result, the colors of the participants' before the occlusion can be used as a measure to match the participants at the end of the occlusion. Therefore color is a distinctive feature between the occlusion participants for the test videos that are used in the analysis and for these videos a color weight above 0.25 is sufficient for the correct matching of occlusion participants. For some other test videos, different values of color weight would be suitable depending on the color difference between the participants.

According to the occlusion handling performance analysis the optimum feature weights are determined for each test video as in Table 5-8.

Table 5-8 Optimum feature weights used in separation score calculation of occlusion handling performance analysis for test videos

Input/Parameter	Size	Color
Test video1	0	0.5
Test video2	0	0.5
Test video3	0	0.5

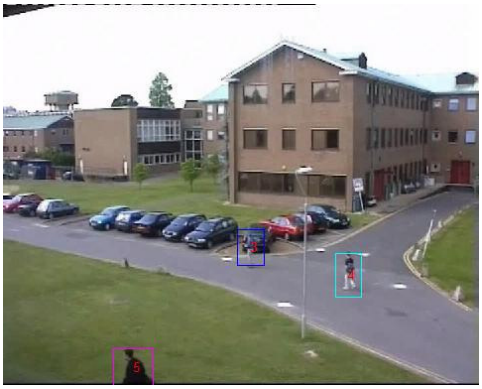
5.4 Results of MHT Algorithm with Other Test Scenarios Using Optimized Feature Weights

In this section, the results of the MHT algorithm are presented with some other test videos using optimized feature weights that are determined in sections 5.2 and 5.3.

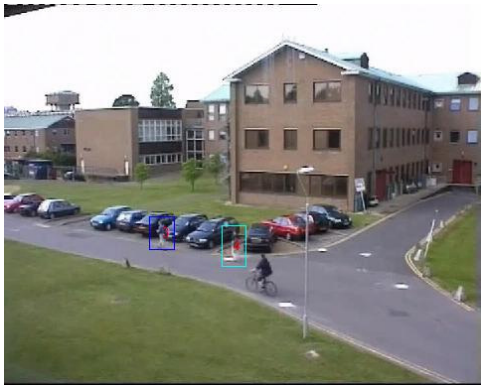
PETS2001/Dataset2/Testing/Camera2 video is run with the optimized feature weights of PETS2001/Dataset1/Testing/Camera1 video and the results of the test are given in Figure 5-9. These two videos have similar features; they are recorded with the same camera and similar outdoor conditions, however they have different targets with different target movements. PETS2001/Dataset2/Testing/Camera2 video has 9 tracks and 3 occlusion cases.

(a)

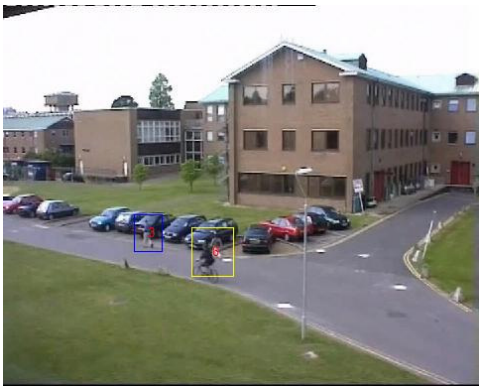
(b)



(c)

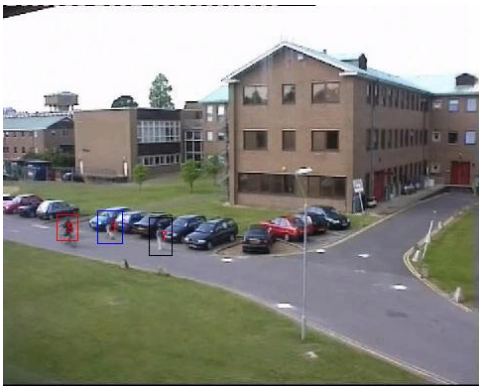


(d)

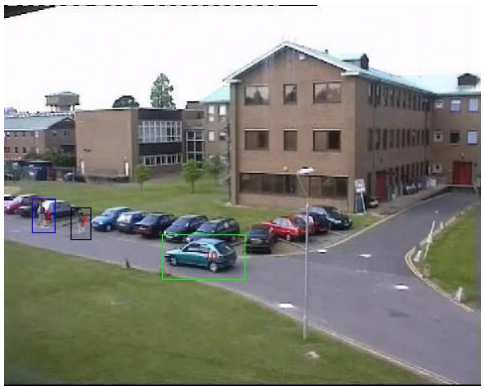


(e)

(f)



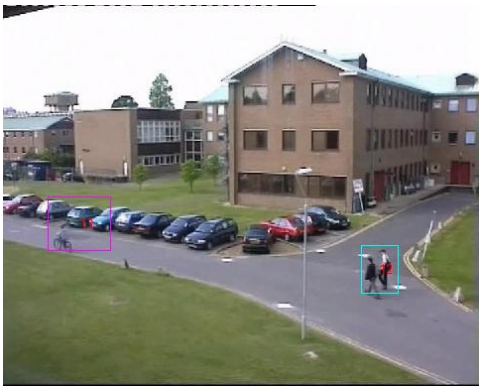
(g)



(h)

(i)

(j)



(k)

(l)

Figure 5-9 Results of PETS2001/Dataset2/Testing/Camera2 video with optimized feature weights of test video1 of section 5.2

The results of the MHT algorithm for PETS2001/Dataset2/Testing/Camera2 with the optimized feature weights show that the algorithm is not successful at tracking the first bicycle-man and handling 3 occlusion cases.

As seen in Figure 5-9 (d), the first bicycle-man is not tracked by the algorithm until he goes far from the camera. Since the bicycle-man moves fast, when he is close to the camera the displacement of him between two consecutive frames is large. Consequently, the observations are not in the gate of the track or the distance score of the hypothesis is low; therefore observation-to-track association is not succeeded. When the bicycle-man goes away from the camera, he becomes a distant track; his displacement between two consecutive frames becomes smaller and as seen in Figure 5-9 (g) he is tracked correctly even he goes fast.

In Figure 5-9, the car is always tracked correctly even it moves fast because the car is a large object and consequently its track gate is large. Thus, there are always observations in its track gate and observation-to-track associations are succeeded.

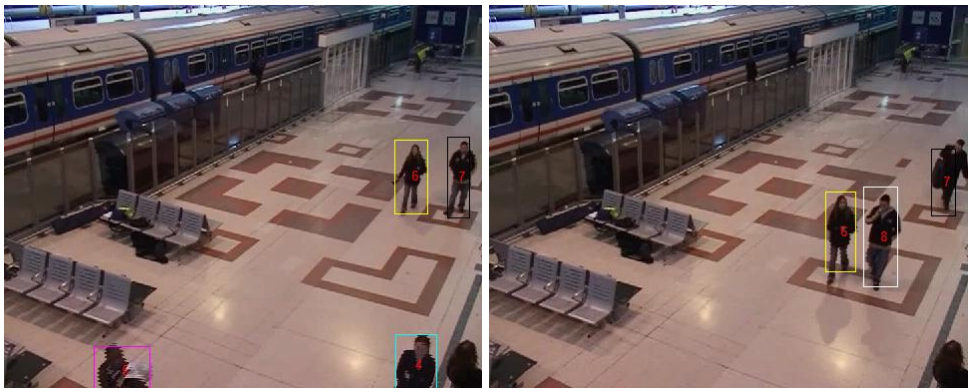
As shown in Figure 5-9 (j) and (l), the second bicycle-man is tracked correctly regardless of the distance between the bicycle-man and the camera because he does not move fast and the displacement between two frames is small enough for correct observation-to-track association.

The occlusions in Figure 5-9 (e) and (f) are not handled correctly because these are not realized as occlusions by the algorithm. The algorithm detects an occlusion when two or more tracks get closer to each other so that they are detected as one object. However, in these occlusion cases, there are not two or more tracks that get closer since the second participant (the bicycle-man) is not a track yet. Therefore the occlusion condition is not satisfied for the cases in Figure 5-9 (e) and (f). In Figure 5-9 (g), track3 is recognized correctly by the algorithm as the bicycle-man goes away from it, however in Figure 5-9 (f), track4 is not recognized correctly and it is labeled as track7. This is because during the occlusion of track4 and the bicycle-man, the displacement of track4 is large; consequently the last position of the dead track (track4) and the position of the new track are far from each other. As a result, the new track is not recognized as track4; instead it is labeled as track7.

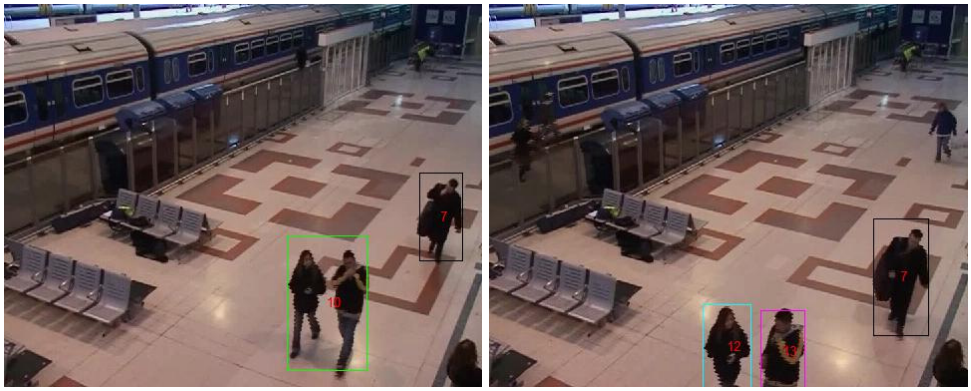
At the end of the occlusion in Figure 5-9 (k), one of the occlusion participants (track10) is not recognized correctly and labeled as track8 as shown in Figure 5-9 (l). The reason of this wrong labeling is the mismatch of the participants with the dead tracks in the occlusion area. At the end of the occlusion, using the separation score, the tracks that are initiated in the thresholded occlusion area are compared with the occlusion participants. Since the separation score of the second occlusion participant (track 10) is above the threshold, this track is not recognized as track10 at the end of the occlusion and it is compared with the tracks which were dead in that area. According to this comparison which is detailed in section 4.2.8.1, this track is matched with the dead track track8.

PETS2006/S2-T3-C video is run with the optimized feature weights of PETS2006/S4-T5-A video and the results of the test are given in Figure 5-10. These two videos have similar features; they are recorded with the same camera and similar indoor conditions; however they have different targets with different target movements. This PETS2006/S2-T3-C video has 11 tracks and 1 occlusion case.

(a) (b)



(c) (d)



(e) (f)

(g) (h)

Figure 5-10 Results of PETS2006/S2-T3-C video with optimized feature weights of test video3 of section 5.2

The results of the MHT algorithm for PETS2006/S2-T3-C video with the optimized feature weights shows that the algorithm is not successful at tracking track7 and track8 and handling an occlusion.

In Figure 5-10 (d), track7 and track8 are labeled wrongly. Since the displacement of track7 (the track labeled as track7 in Figure 5-10 (c)) is large between two consecutive frames, observation-to-track association is not succeeded, track7 (the track labeled as track7 in Figure 5-10 (c)) is dead and a new track called track8 is initiated in Figure 5-10 (d). The track labeled as track7 in Figure 5-10 (d) should be labeled as a new track however, this track is matched with a track which was dead in that area because of the similarity of their position, size and color and therefore labeled as that dead track.

At the end of the occlusion in Figure 5-10 (e), occlusion participants are not recognized correctly and labeled as track12 and track13 as shown in Figure 5-10 (f). Since the occlusion participants (track6 and track8 of Figure 5-10 (d)) are close tracks (close to the camera) and they are moving towards the camera, the differences in the occlusion participants' position between two consecutive frames are large. Consequently, the new tracks that are formed after the separation of the occlusion participants (track12 and track13) are not in the thresholded occlusion area where the occlusion ends. Therefore, these new tracks are not compared with the old occlusion participants (track6 and track8) and as a result the occlusion participants are not recognized at the end of the occlusion; instead they are labeled as track12 and track13.

CHAPTER 6

CONCLUSION

6.1 Summary and Conclusions

Visual surveillance systems are mostly used in airports, train stations, banks and country borders as a solution to security problems. Since automated surveillance systems consist of cameras and computers, the performance of these systems increases as computer technology develops. In this study, moving object detection and moving object tracking parts which are the basic blocks of visual surveillance systems have been implemented and the overall performance of the developed system is analyzed.

The first part of the developed visual surveillance system is moving object detection in which the detection of foreground objects is performed. Foreground segmentation and sub-operations are the sub-steps of this part. Mixture of Gaussians method has been utilized for foreground segmentation. This method is an adaptive model and robust to dynamic changes in the environment. Therefore, this method is suitable for environments with gradual lightning changes, repetitive motions like swaying of trees. The duration of the background training period has an effect on the detection performance and therefore on the performance of the whole system. It is observed that the performance of the system is better if background training period is longer and the learning rate is relatively small in the training period since the background is learned well. After the detection of moving objects, sub-operations step is performed which consists of shadow removal and morphological

operations steps. A statistical method is applied for shadow removal. Shadow removal is important since they may cause false segmentation of object features in terms of their shape and size and furthermore, large shadows may cause false occlusions. After shadow removal, morphological operations like erosion and dilation are performed on detected objects. The number and order of erosion and dilation operations and also the size and the shape of the structuring elements may be different for different input videos. In this study, three dilations, after two erosion operations are implemented with same structuring element. Morphological operations are followed by connected component labeling.

The second part of the developed visual surveillance system is moving object tracking which is implemented using multiple hypothesis tracking method. The performance of this method is good at tracking multiple targets since this method generates hypotheses for the solution of the observation-to-track association problem and defers the decision to the following frames rather than deciding the solution with the current data. Since the deferred decision is made using the data of more than one frame, more reliable assignments are made with this method. In hypothesis evaluation, track features such as distance, size, color, gate size and track history are utilized. In addition, this method is capable of implementing track initiation, confirmation and deletion. However, its computational load and the difficulties in constructing the track tree structures are the disadvantages of this method. Track confirmation and occlusion handling are also realized in this study. Size and color features of the occlusion participants are utilized for occlusion handling.

The performance of the system is examined with three analyses. The first analysis aims to understand the effects of the duration of background training on the overall tracking performance. It is observed that 35 – 40 seconds long background training period with a relatively small learning rate via the tracking part results in high tracking performance because of the well-defined background. The second analysis presents the sensitivity of the tracking performance with respect to the features that are used for hypothesis score calculation. According to this analysis, the algorithm

is not sensitive to the changes in the weight of the distance and size while tracking distant objects whereas it is sensitive in close objects case; the algorithm is not sensitive to the changes in the weight of the color feature when tested with indoor videos whereas it is sensitive with outdoor videos and the algorithm is sensitive to the changes in the gate size and history weights in any condition. The third analysis presents the sensitivity of the occlusion handling performance with respect to the features that are used in separation score calculation. According to this analysis, the algorithm is not sensitive to the changes in the weight of the size feature, whereas, it is sensitive to the changes in the weight of the color feature considering the occlusion handling performance for all the test videos used in the analysis.

6.2 Future Work

The proposed tracking system is designed for a single camera case. Using more than one camera can be helpful to improve the occlusion handling performance. In addition to occlusion handling, the performance of tracking for targets passing behind an object or leaving the field of view of the camera can also be improved. However, it brings the additional problem of information fusion. This problem can be solved by implementing a supervisory system to deal with data fusion. The homography between the views of the cameras can be utilized for this purpose. According to the performance analysis the system has sensitivity to the parameters used in hypothesis score calculation and occlusion handling. The sensitivity of the system can be reduced by implementing a supervisory system that automatically tunes these parameters according to the features of the tracks such as being close to the camera. Another improvement of the system may be using pan-tilt-zoom cameras instead of static ones. This may be helpful for tracking targets leaving the field of view or observing the motion of the targets in more detail by zooming if necessary. For future MHT implementations using a structure based programming language which is capable of implementing tree structure would be helpful in implementing track tree forming and tree pruning.

REFERENCES

[1] M.F. Abdelkader, R. Chellappa and Q. Zheng, “Integrated Motion Detection and Tracking for Visual Surveillance”, Fourth IEEE International Conference on Computer Vision Systems, pp. 28-28, January 2006.

[2] Z. Chaohui, D. Xiaohui, X. Shuoyu, S. Zheng, L. Min, “An Improved Moving Object Detection Algorithm Based on Frame Difference and Edge Detection”, Fourth International Conference on Image and Graphics, pp. 519-523, August 2007.

[3] Yuri Ivanov, Chris Stauffer, Aaron Bobick, W.E.L. Grimson, "Video Surveillance of Interactions", Second IEEE Workshop on Visual Surveillance, pp. 82, 1999.

[4] K.S. Knudsen and L.T. Bruton, “Moving Object Detection and Trajectory Estimation in the Transform/Spatiotemporal Mixed Domain”, International Conference on Acoustic Speech and Signal Processing, vol. 3, pp. 505-508, Mar 1992.

[5] H. Yang, Y. Tan, J. Tian and J. Liu, “Accurate Dynamic Scene Model for Moving Object Detection”, International Conference on Image Processing, vol. 6, pp. 157-160, September 2007.

[6] C. Zhao, W. Liu, Y. Wang, Y. Cheng and H. Zhang , “A Fast Algorithm for Moving Objects Detection Based on Model Switching”, International Conference on Audio, Language and Image Processing, pp. 143-146, July 2008.

[7] L. Maddalena and A. Petrosino, “Moving Object Detection for Real-Time Applications”, 14th International Conference on Image Analysis and Processing, pp. 542-547, September 2007.

[8] H.C. Zeng and S.H. Lai, “Adaptive Foreground Object Extraction for Real-time Video Surveillance with Lighting Variations”, International Conference on Acoustic Speech and Signal Processing, vol. 1, pp. 1201-1204, April 2007.

[9] P.D.Z. Varcheie, M.S. Lavoie and G.A. Bilodeau, “An Efficient Region-Based Background Subtraction Technique”, Computer and Robot Vision, 2008. CRV'08. Canadian Conference, pp. 71-78, May 2008.

[10] W. Kim and J.J. Lee, “Visual Tracking using Snake for Object’s Discrete Motion”, International Conference on Robotics and Automation, pp. 2608-2613, May 2001.

[11] P.L. Rosin and B. Friedlander, “Image Difference Threshold Strategies and Shadow Detection”, British Conference on Machine Vision, vol. 1, pp. 347-356, 1995.

[12] A.J. Lipton, H. Fujiyoshi and R.S. Patil, “Moving Target Classification and Tracking from Real-time Video”, 4th IEEE Workshop on Applications of Computer Vision, October 1998.

[13] S.J. McKenna, S. Jabri, Z. Duric, H. Wechsler and A. Rosenfeld, “Tracking Groups of People”, The Fourth International Conference on Automatic Face and Gesture Recognition, Grenoble, France, March 2000.

[14] N. Paragios and R. Deriche, “Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 3, pp. 266-280, March 2000.

[15] M. Kass, A. Witkin and D. Terzopoulos, “Snakes: Active Contour Models”, International Journal of Computer Vision, vol. 1, pp. 321-331, 1988.

[16] T. Horprasert, D. Harwood and L.S. Davis, “A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection, 7th IEEE International Conference on Computer Vision, Frame Rate Workshop, pp. 1-19, May 1999.

[17] A. Prati, I. Mikic, M.M. Trivedi and R. Cucchiara, “Detecting Moving Shadows: Algorithms and Evaluation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 7, pp. 918-923, July, 2003.

[18] S.S. Blackman, “Multiple Hypothesis Tracking for Multiple Targets”, IEEE Aerospace and Electronic Systems, vol. 19, no.1, Part 2: Tutorials, January 2004.

[19] 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.

[20] I.J. Cox and S.L. Hingorani, "An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking" IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 2, pp.138-150, February 1996.

[21] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time Tracking of the Human Body," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 780-785, 1997.

[22] 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.

[23] E. Salvador, Andrea Cavallaro and T. Ebrahimi, "Shadow Identification and Classification Using Invariant Color Model", IEEE International Conference on Acoustics, Speech and Signal Processing, Lake City, UT, vol.3, pp. 1545-1548, May 2001.

[24] D.B. Reid, "An Algorithm for Tracking Multiple Targets," IEEE Transactions on Automatic Control, vol. 24, no. 6, pp. 843-854, December 1979.

[25] S. Blackman and R. Popoli, "Design and Analysis of Modern Tracking Systems", Artech House Radar Library, Boston, 1999.

[26] İ. Haritaoğlu, 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, August 2000.

[27] Y. XiaHou, S. Gong, "Adaptive Shadows Detection Algorithm Based on Gaussian Mixture Model", International Symposium on Information Science and Engineering, vol. 1, pp. 116-120, December 2008.

[28] R. Polana and R. Nelson, “Low Level Recognition of Human Motion”, IEEE Workshop on Motion of Non-rigid and Articulated Objects, International Symposium on Information Science and Engineering, pp. 77-82, November 1994.

[29] I.J. Cox and J.J. Leonard, “Probabilistic Data Association for Dynamic World Modeling: A Multiple Hypothesis Approach”, Fifth International Conference on Advanced Robotics, vol.2, pp. 1287-1294, June 1991.

[30] University of Reading, PETS 2001, http://ftp.pets.rdg.ac.uk/PETS2001/, September 2009.

[31] University of Reading, PETS 2006, http://ftp.pets.rdg.ac.uk/PETS2006/, September 2009.

[32] F. Meyer and P. Bouthemy, “Region-Based Tracking Using Affine Motion Models in Long Image Sequences”, CVGIP: Image Understanding, vol. 60, pp. 119-140, 1994.

[33] O. Akman, “Multi-camera Video Surveillance: Detection, Occlusion Handling, Tracking and Event Recognition”, Master Thesis, Middle East Technical University, August 2007.

[34] K.G. Murty, “An Algorithm for Ranking All the Assignments in Order of Increasing Cost”, Operations Research, vol. 16, pp. 682-687, 1968.

[35] 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.

[36] W. Zhang, X. Fang and X. Yang, "Spatiotemporal Gaussian Mixture Model to Detect Moving Objects in Dynamic Scenes", Journal of Electronic Imaging, vol. 16, no.2, pp. 023013, May 2007.

[37] M. Yılmaz, “Multiple Target Tracking Using Multiple Cameras”, Master Thesis, Middle East Technical University, May 2008.