A VIDEO SURVEILLANCE SYSTEM BASED ON INTERACTING MULTIPLE MODELS

by
Ceren ÖRS SARITAŞ

Submitted to the Institute of Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Electrical & Electronics Engineering

Yeditepe University
2010

A VIDEO SURVEILLANCE SYSTEM BASED ON
INTERACTING MULTIPLE MODELS

APPROVED BY:

Assoc. Prof. Dr. Cem Ünsalan ……..
(Thesis Supervisor)

Assoc. Prof. Dr. Kemal E. Tepe ……..

Asst. Prof. Dr. Duygun E. Barkana ……..

DATE OF APPROVAL:

ACKNOWLEDGEMENT

I would like to thank to my thesis supervisor, Assoc. Prof. Dr. Cem Ünsalan, for his guidance, instructive comments, suggestions and patience throughout the development of this thesis. I appreciate helpful comments given by my thesis committee members Assoc. Prof. Dr. Kemal E. Tepe and Asst. Prof. Dr. Duygun E. Barkana. Moreover, I want to thank my colleagues Dr.Köksal Hocaoğlu who introduced me to the field of image processing and Mehmet Yılmaz for his inspiring discussions. Finally, I am especially grateful to my dear husband I. Engin Sarıtaş and my family for the happiness they brought to my life.

ABSTRACT

A VIDEO SURVEILLANCE SYSTEM BASED ON INTERACTING MULTIPLE MODELS

The video-based surveillance systems are becoming widespread due to the increasing security needs. Consequently, these systems bring huge volumes of visual data to be analyzed. The automated systems are developed to assist human operators in time-consuming scene analysis. Besides, they enhance the surveillance efficiency by tracking interesting moving objects and interpreting the tracking results for potentially dangerous situations or suspicious activities.

In this thesis, we present an automated visual surveillance system with real-time and robust tracking capabilities. The system detects moving objects under changing background conditions by an adaptive Mixture of Gaussians method. The detected objects in the consecutive video frames are properly associated with each other by means of data validation and association algorithms. The tracking algorithm makes use of the prediction and estimation results of the Interactive Multiple Modal (IMM) estimator operating on constant velocity and coordinated turn motion models simultaneously. The system has been used to analyze PETS 2001 datasets which provide a unique test environment for the objective evaluation of the tracking algorithms.

ÖZET

ETKİLEŞİMLİ ÇOKLU MODELLERE DAYALI VİDEO GÖZETİM SİSTEMİ

Video tabanlı gözetim sistemleri artan güvenlik ihtiyaçları sebebiyle giderek yaygınlaşmaktadır. Bunun sonucunda, bu sistemler çok büyük miktarlarda kayıtlı görüntü veri analizi ihtiyacını beraberinde getirmektedir. Yazılım destekli sistemler, zaman alan görüntü analizinde operatörlere yardım etmek üzere geliştirilmektedir. Ayrıca, bu sistemler hareketli nesneleri takip ederek, takip sonuçlarını olası tehlikeli durumlar ve şüpheli aktiviteler için yorumlayarak gözetim verimliliğini arttırlar.

Biz bu tezde, gerçek zamanlı ve güvenilir takip yeteneklerine sahip bir yazılım destekli gözetim sistemi sunuyoruz. Sistem, hareketli nesneleri değişken arka plan koşullarında adaptif bir Gauss'ların karışımı yöntemiyle tesbit eder. Tesbit edilen nesneler, ardışıl video karelerinde veri doğrulama ve eşleştirme algoritmalarıyla uygun olarak birbirleriyle eşleştirilir. Takip algoritması, sabit hız ve dönüş hareket modelleri üzerinde aynı anda çalışan etkileşimli çoklu model (IMM) kestiricisinin tahmin ve kestirim sonuçlarından yararlanır. Bu sistem, takip algoritmalarının tarafsız olarak değerlendirilmesine imkan veren özel bir takip ve gözetim performans değerlendirmesi (PETS 2001) veri kümesi üzerinde denenmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

A	State transition matrix
E	Expectation function
F	System matrix
H	Measurement matrix
K	Number of Gaussian distributions in the mixture
L	Noise gain
N	Gaussian pdf
p	pdf
P	Estimate error covariance matrix
Pr	Probability of an event
R	Measurement noise matrix
Q_c	Power spectral density of a zero-mean Gaussian white noise process
q_k	Zero-mean Gaussian white noise sequence
Q_k	Covariance of q_k
Q	Constant Q_k matrix
T	Transposition (of a matrix or a vector)
X_t	Pixel value at sampled frame t
$\hat{x}$	Estimate of state x
Z^+	Non-negative integers
$\hat{z}$	Predicted measurement
-1	Inverse of a matrix
α	Significance level
∇_x	Gradient with respect to (the vector) x
$\delta[n]$	Kronecker discrete delta function (defined in Subsection 5.2.1)
$\delta(\tau)$	Dirac continuous delta function (defined in Subsection 5.2.1)
χ^2	Chi-square distribution with n degrees of freedom
$\forall$	For all

$\lvert . \rvert$	Determinant (of a matrix) or magnitude (of a scalar)
arg min	Argument that minimizes
cov	Covariance function
dim	Dimension of a vector
exp	Exponential function
ln	Natural logarithm function
CWNA	Continuous White Noise Acceleration
IEEE	Institute of Electrical and Electronics Engineers
IMM	Interacting Multiple Model (estimator)
LTI	Linear Time Invariant
MATLAB	High level computing language developed by Mathworks
MMSE	Minimum Mean Square Error
MSE	Mean Squared Error
NN	Nearest Neighbor
PETS	Performance Evaluation of Tracking and Surveillance
pdf	probability density function
RAM	Random Access Memory
var	Variance function

1. INTRODUCTION

Video-based surveillance systems started with analog closed-circuit television (CCTV) systems in order to monitor security-sensitive areas. These systems consist of a number of cameras connected to a set of monitors through automated switches [1]. As the technology evolved, deployment of surveillance cameras become widespread. Eventually, human supervision turned out to be expensive and inadequate to ensure proper monitoring. Classical visual surveillance systems lost their primary benefit as an active, real-time medium [2]. In common practice, the surveillance cameras are monitored sparingly by human operators and the recorded videos are used as a forensic tool to make investigation after an abnormal event has taken place.

Next, automated video surveillance systems are developed to assist human operators in time-consuming scene analysis. Instead of passively recording the footage, they analyze the video streams from a single or multiple cameras by means of software. The analysis consists of detecting interesting moving objects, tracking these objects frame by frame and interpreting the tracking results. These systems can also be designed to detect potentially dangerous situations or suspicious activities as they happen. Besides, they can also take appropriate actions like alerting a human operator and/or starting to record the incoming images. As a result, they increase both the efficiency and the security.

These automated systems are especially needed for public and military security purposes. They are also proposed in many other application domains like smart video data mining, congestion analysis of the people and vehicles, traffic monitoring, etc. The public places like airports, museums, stations, parking lots, banks, shopping centers, and buildings have long been monitored by surveillance cameras. However, automated systems can detect suspicious objects (such as a suitcase left in an airport lounge) or a burglary in progress. In addition, these systems can track suspects over a wide area by using the cooperation of multiple cameras [3]. For the military security purposes, automated systems can track military targets like aircrafts, missiles, land vehicles or monitor national borders.

1.1. Overview

In this thesis, an automated visual surveillance system with moving object detection and tracking capabilities is proposed. Because of the support role in events, automated video surveillance systems have to operate in real-time [4]. This means processing of a single frame should be done at a rate equal to or faster than the video frame rate. In the proposed system, algorithm selections and design criterions are considered to meet the real-time constraints. The proposed system processes each incoming video frame recursively. One-cycle of the algorithm can be divided into four steps as illustrated in Figure 1.1.

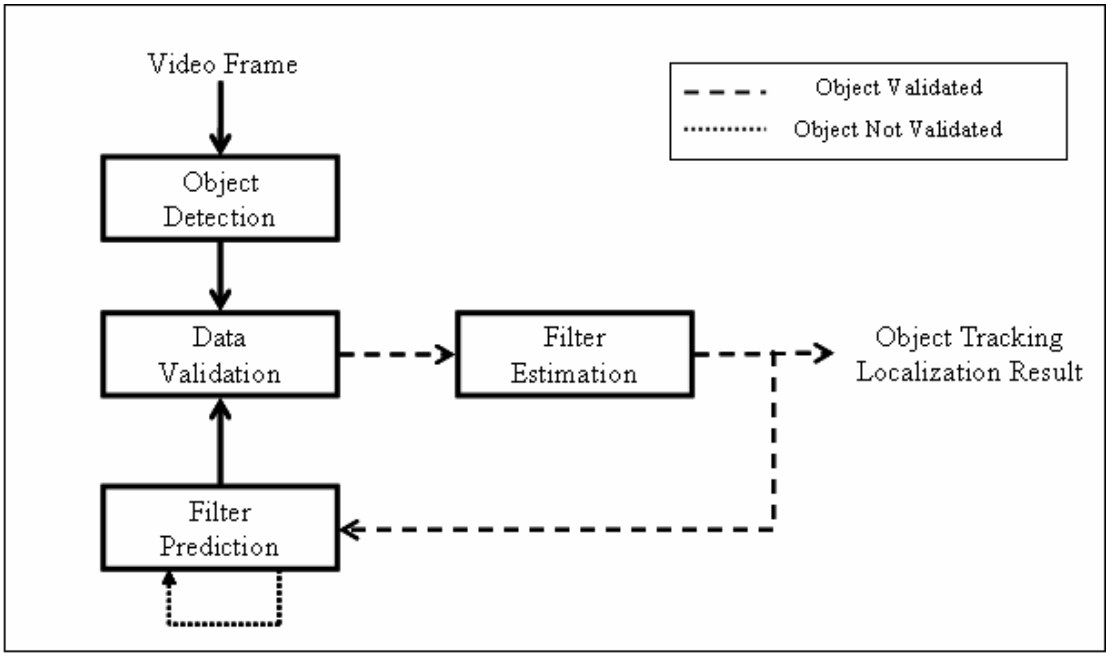


Figure 1.1. An overview of the proposed surveillance system

In the object detection step, the moving objects are detected by Mixture of Gaussians [5] method. The data validation, filter prediction and estimation steps are processed for each detected moving object. The detected objects in the consecutive video frames are associated with each other by means of data validation and association algorithms. Data validation algorithm determines the probable objects in the current frame to associate with the tracked objects. When there is more than one probable object for one tracked object,

the Nearest Neighbor (NN) [6] data association algorithm selects the most relevant one. The data association step is not illustrated in Figure 1.1.

Object tracking can be described as the process of determining the locations of the moving objects in every frame of a given video stream. The moving object can be located as a whole or some extracted features/parts of it depending on the application. The proposed tracking algorithm tracks the centroids of moving objects in successive frames using Interactive Multiple Modal (IMM) [7] filtering technique. First, the filter makes a prediction about the position of the centroid in the image plane. This prediction is compared with object detection results in the data validation step. The validated object centroid position is used in the filter in order to estimate the true location of the target in the image plane. If there is no validated position for the tracked object, the dedicated filter only makes a prediction about its location in each cycle of the algorithm until a validated position arrives.

Kalman filter [8] uses a single motion model to estimate the location of the tracked object in the image plane. On the other hand, IMM filter combines several motion models appropriately. The reason for using IMM filter is that one motion model inadequately represents the whole motion pattern of moving objects. Finally, the median-based object detection and mean-shift tracking algorithms are also implemented for the comparison of detection and tracking performances, respectively.

1.2. Organization of the Thesis

The remaining part of this thesis is organized as follows: Chapter 2 describes the main ideas behind recent research in moving object detection methods used in video surveillance applications, with emphasis on the Mixture of Gaussians method. Chapter 3 provides a brief summary of the tracking approaches proposed in the literature. The problems encountered in detection and tracking steps are also presented in relevant sections of Chapter 2 and Chapter 3, respectively. Data validation and association algorithms are described in Chapter 4. Chapter 5 provides the background information about Kalman filters. Kalman and IMM filtering methods are described in Chapter 6 and

Chapter 7, respectively. Chapter 8 presents the experimental results of the proposed system. Finally, Chapter 9 concludes the thesis with suggestions for future research.

2. OBJECT DETECTION

Object detection (or motion segmentation) algorithms segment out regions corresponding to moving objects in image sequences. These regions provide a focus of attention because only these regions need to be considered in subsequent processes such as tracking and behavior analysis [3].

Most motion segmentation methods build a background model which is a representation of the scene without any moving object in it. Then, moving objects are segmented by thresholding a per-pixel distance between the current frame and the background model [9]. If the per-pixel distance is computed by subtracting the pixel values of each new frame from the corresponding pixel values of the background model, this technique is known as background subtraction. Here, pixels in which the distance is above a threshold belong to the object of interest and are classified as “foreground”. The result of the motion segmentation algorithm is a binary image where the background pixels are assigned zero and the foreground pixels are assigned one. This binary image is generally referred as “foreground image”.

The connected foreground pixels constitute a region called blob in the foreground image. Morphological post processing operations are typically applied to fill the small holes inside the foreground regions (blobs) and to eliminate very small-sized, noisy blobs. Due to the dynamic nature of scenes, common problems in object detection step are changing illumination levels, shadows and multi-modal background colors. The next section describes these problems and illustrates their consequences in the foreground images.

2.1. Problems of Background Modeling

Most background modeling methods are based on the color characteristics of the scene. Because of the changes in lighting conditions, the colors of the scene alter continuously. For example, in an outdoor scene moving clouds in the sky change

illumination levels. Similarly, in an indoor scene opening of doors or windows change illumination levels [1]. Figure 2.1 illustrates a scene undergoing an illumination change and corresponding motion segmentation results [10].

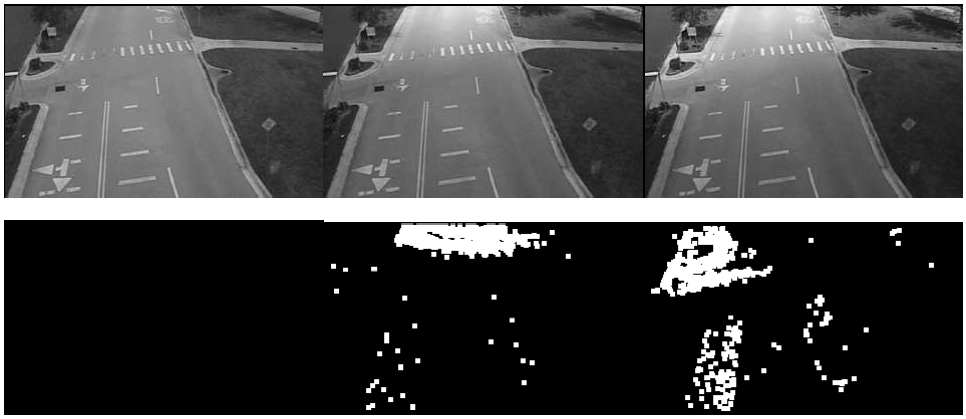


Figure 2.1. A scene undergoing an illumination change

In Figure 2.1, the number of falsely detected foreground regions increase as the illumination changes. Therefore, a robust background model must be adaptive to illumination changes. Otherwise, illumination changes are falsely detected as foreground regions.

An adaptive model learns the background at some pre-defined learning rate. Learning is accomplished by recursively updating the background model with new incoming video frames. The learning rate is an important design parameter. However, the method has to find a compromise between two conflicting demands. On one hand, learning rate should be high enough to deal with changes in illumination. On the other hand, learning rate should be low enough to avoid learning slowly moving objects as background [11]. The second problem occurs if the color properties of moving objects are very similar to the background. Then, it might not be possible to distinguish between the two. An example is shown in Figure 2.2 [10]. In this figure, the person at the top right of the image (highlighted by a bounding box) could not be detected properly.

Figure 2.2. A camouflaged person

Another problem is shadows. When objects cast shadows, their exact size and shape could not be determined correctly. For instance, a person casting a shadow is illustrated in Figure 2.3 [10]. Also, the shadow may touch a nearby object causing the algorithm to consider the two separate objects as one [4].

Figure 2.3. A person casting a shadow

In addition to the above problems, during the visualization process, a particular pixel may represent a single or multiple background colors [5]. Multiple colors can be the result of repetitive background motion, reflectance or shadows. As an example, as illustrated in Figure 2.4, swaying branches and leaves on a tree in front of a road will cause the same pixel location to represent values from tree leaves, tree branches, and the road itself [10]. Moving clouds, small camera displacements, ripples on water are also good examples for repetitive background motion.

Figure 2.4. A tree swaying in the wind

Finally, if a previously moving object becomes motionless, it should be considered as background. Conversely, when a motionless object starts to move, not only itself but also revealed parts of the new background are usually detected (perceived) as a moving region. However, in this case the new background should be learned as quickly as possible.

2.2. Previous Work

Background models are built by using the information extracted from images. Color information is the most common one. Here, pixels of the background model are represented as either having a single color value or a probabilistic distribution of different color values. A conventional approach for background modeling is taking sample images and computing the cumulative average (arithmetic mean) of the color values for each pixel location. Recent studies point out that the median value is far more robust than the mean value [12]. [13] proposed to use the median value of the previous N (typically between 50 and 200) frames for each pixel location in the background model. They observed that color images give better segmentation results than grayscale images especially in low contrast areas such as objects in dark shadows.

If the difference between the reference background image pixel $B(x,y)$ and the current image pixel $I(x,y)$ is above a certain threshold T, then the pixel is marked as foreground,

$$\sum_{C\in\{R,G,B\}} \left| I_C(x,y) - B_C(x,y) \right| > T \tag{2.1}$$

[13] compute the threshold value as the product of average standard deviation of the white noise in the video system for the color channels and some heuristically selected constant. The median-based approach is computationally simple but requires the recent pixel values of past $N-1$ frames. In general, these simple methods are not robust in segmentation problems mentioned before. They are very sensitive to illumination changes. Hence, they can only work in nearly static background scenes.

To improve the accuracy of the segmentation process, some methods use color features along with other features of the scene. The rationale here is that, features like edge and texture do not vary much with illumination changes as compared to color features [14]. For example, [15] utilized both the color and edge information to build background models in two separate parts; one is a color model and the other is an edge model. Background subtraction is performed for each model and their results are combined to find foreground regions. [16] integrated both the color and texture differences between two frames. Some object detection methods make use of estimation techniques. In [17], Kalman filtering is used for adaptive background model estimation. The approach takes into account the changing illumination and the problems caused by the slow or non-continuously moving (temporarily stationary) foreground objects.

More advanced background models use the statistical characteristics of pixel values. [18] built a statistical model of the stationary background. They observe the color values at each pixel position in consecutive frames and associate them with a single Gaussian distribution. Then, for each incoming frame and each pixel position, the likelihood of its color coming from associated Gaussian is computed. The pixel which deviates from the distribution (background model) is labeled as a foreground pixel [14]. Otherwise, it is used to update the mean and covariance of the distribution to compensate for illumination changes.

These methods are less sensitive to illumination changes. But a robust background modeling method should also consider multiple-color backgrounds. The method of Mixture of Gaussians handles multiple-color backgrounds with simple computations. [5] used a mixture of Gaussians to represent the color values of each pixel. Some of the distributions correspond to the background model and the others to the foreground model. Therefore, each pixel constructs its own background model. It is classified as foreground or background depending on whether its new value matches with one of the background model distributions. Distribution parameters are recursively updated at each incoming video frame.

We used the Mixture of Gaussians method in our visual surveillance system. It robustly deals with lighting changes, slowly moving objects and multiple-color backgrounds. A detailed description of the method is given in the next section.

2.3. Mixture of Gaussians

In [5], Mixture of Gaussians is proposed to represent an adaptive multiple-color background model per pixel. In this method, the color value of each pixel is separately modeled by the mixture of K Gaussian distributions. Each k^{th} distribution in the mixture models a different color with mean μ_k and covariance $\sum_k$. Covariance can be thought as a measure of dispersion of that color value around its mean. The time proportions that those colors stay in the scene are represented by weight parameters w_k [19]. Weights are normalized as

$$\sum_{k=1}^{K} w_k = 1 \tag{2.2}$$

Let the observed value of a particular pixel at sampled frame t be denoted by X_t. In our project, X_t is one-dimensional (monochrome intensity) but it can be n-dimensional depending on the color space chosen. The probability to observe X_t within all the previous values of that pixel is determined as

$$\Pr\{X_t\} = \sum_{k=1}^{K} w_{k,t} N(X_t; \mu_{k,t}, \Sigma_{k,t}) \tag{2.3}$$

Naturally, the mixture not only models multiple-color backgrounds but also moving objects in the scene. However, the statistical properties of Gaussians representing the foreground are different from those of the background. Particularly, the background values will occur more frequently (with high weights w_k) and do not vary much (having small variances σ_k) [9] .

When the distributions are ordered according to their fitness values w_k / σ_k , in decreasing order, the distribution with the highest fitness value is the most probable background distribution for that pixel [19]. The others are less probable transient background distributions or foreground. Therefore, the first B of the ordered distributions are used as a model of the background scene as

$$B = \arg\min_b \left(\sum_{k=1}^{b} w_k > T \right) \tag{2.4}$$

where, a pixel will represent a background value in the $100T\%$ portion of time.

In order to classify pixels as foreground (moving regions) or background, the pixel value X_t is checked against the K Gaussian distributions, in the order of fitness, until a match is obtained [5]. A match is defined as a pixel value within 2.5 standard deviations of the distribution's mean value [5]. If X_t matches with one of the pixel's B background distribution(s), then it is considered a background pixel. The 2.5 standard deviations criterion serves as per pixel/per distribution threshold [5].

The background model adapts to changes in illumination and runs in real-time by an updating procedure [19]. The first matched Gaussian is updated according to the following update equations

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

$$\sigma_{k,t}{}^2 = (1-\rho_{k,t}) \sigma_{k,t-1}{}^2 + \rho_{k,t} (X_t - \mu_{k,t})^T (X_t - \mu_{k,t}) \tag{2.6}$$

$$\mu_{t,k} = (1-\rho_{k,t}) \mu_{k,t-1} + \rho_{k,t} X_t \tag{2.7}$$

In this thesis, we computed the $\rho_{k,t}$ term in Equations 2.6 and 2.7 different from the original work [5]. According to [9], a faster and more logical computation for $\rho_{k,t}$ is as

$$\rho_{k,t} \approx \frac{\alpha}{w_{k,t}} \tag{2.8}$$

As a result of these equations, the matched distribution parameters are changed: the mean value is moved in the direction of the pixel value, the weight is increased and the variance is decreased. α is the learning constant which controls the amount of change i.e. how much the current pixel value influences the background model.

The parameters for unmatched Gaussians remain the same except the weight parameter which is decreased proportional to the learning rate as

$$w_{k,t} = (1-\alpha) w_{k,t-1} \tag{2.9}$$

If X_t matches none of the K distributions, the least probable (lowest fitness valued) distribution is replaced by a new Gaussian centered at X_t with an initially high variance and low prior weight [5].

One of the most useful properties of this update scheme is that, it handles the situations arising from temporarily stopped objects. If a previously moving object becomes stationary long enough, it will gradually become part of the background model. However, according to Equation 2.9, the distribution describing the original background still remains in the mixture with the same mean μ_k and variance σ_k, but with a lower weight w_k. When the object starts moving again, the original background pixel values reappears in the scene and quickly incorporates into the background model [5].

A practical minimum value is $K = 3$ in order to model two-colored background and one foreground in each pixel. Only one foreground Gaussian will be enough because it can be used roughly to model any foreground pixel [9]. It has been reported that not much improvement is obtained beyond $K = 5$.

The resulting binary segmented image can be refined by standard morphological functions. In this thesis, we eliminate very small noise-generated blobs with simple area thresholding and set a pixel to 1 if five or more pixels in its 3-by-3 neighborhood are 1's; otherwise, set the pixel to 0. An example of Mixture of Gaussian modeling for background subtraction is given in Figure 2.5 [14].

Figure 2.5. A person is walking across the scene

The image in the first column of Figure 2.5 is taken from a sequence in which a person is walking across the scene. The second column represents the mean value of the highest-weighted Gaussian at each pixel position. These values represent the stationary background. The third column is the mean value of the second-highest weighted Gaussian at each pixel position. These values represent colors which are observed less frequently. The last column is the result of background subtraction. The foreground consists of pixels in the current frame that does not match a background Gaussian distribution.

3. OBJECT TRACKING

In visual surveillance applications, the object tracking phase generates the trajectory of a moving object by locating its position in every frame of a given video stream [14]. Algorithms start the tracking of new objects when they enter the scene and terminate the tracking of existing objects when they exit the scene.

As in the object detection phase there are also some problems encountered in object tracking. For instance, in some frames the tracked object could not be located properly because of the occlusion problems. A moving object can be occluded by stationary scene structures like trees or buildings. Also, when moving objects come very close to each other, they form a single moving region in the foreground image. Thereby, they occlude each other. Since the view of an occluded object is partially or fully blocked, a robust tracking algorithm should be able to predict its possible locations in the image plane.

Another problem is the appearance change of tracked objects. Objects can change their velocity and direction as they move through a camera's field of view. Even if objects move with constant speeds and directions, their size will change according to relative distance to the camera. Hence, tracking algorithms have to take into account the appearance changes.

The next section describes some of the methodologies which are commonly used in object tracking applications. The methodologies can be classified as region-based, feature-based and boundary-based approaches. However, these approaches can also be combined to improve the accuracy, with the cost of computational complexity.

3.1. Methods of Object Tracking

In this section, the approaches used in object tracking applications are briefly described and a representative work for each of them is given. The mean-shift tracking

method under region-based approach is described in detail. Because, this method is also implemented in this thesis for performance and tracking quality comparisons.

3.1.1. Region-Based Tracking

Once the motion segmentation algorithm segments out the regions corresponding to moving objects, region-based tracking algorithms establish a correspondence between these regions in subsequent video frames [4]. This correspondence is usually based on the similarities between one or more properties of these regions such as size (area), color, shape, velocity, centroid, etc. In our visual surveillance system, we track the centroids of segmented regions.

Region-based algorithms are suitable for real-time processing but their performance highly depends on the motion segmentation step. In [20] a region tracking algorithm by using the mean shift procedure is proposed. The mean-shift tracking algorithm is initialized with the segmented region of the tracked object namely the target region. Segmentation is done only at the beginning of the algorithm. In order to represent (the color properties of) the target region, they build an m-bin histogram using both the color values of the pixels comprising the region and a kernel which weights the pixels according to their distances from the region center.

Tracking is accomplished by iteratively searching a region having the most similar histogram to the histogram which represented the target region in the previous video frame. The similarity between the two histograms is measured by a similarity function derived from the Bhattacharyya coefficient. The local maximum of this similarity function in the image plane indicates the new target location. The iterations are repeated to find the local maximum by using the mean-shift procedure. Mean-shift is an efficient gradient-based optimization method [21]. The iterations start at the location of the target in the previous frame, namely the first candidate location. In each iteration, the mean shift algorithm increases the similarity by shifting the candidate location to a new location. Iterations are repeated until the shifting amount becomes very small.

In this method, the kernel should be isotropic (differentiable) with a convex and monotonic decreasing kernel profile like Gaussian or Epanechnikov kernels. The reason is two-fold; first, this kernel profile gives smaller weights to the pixels farther from the region center [22]. So, the peripheral pixels are assigned the smallest weight because they are the least reliable, being often affected by occlusions. Second, a differentiable kernel profile yields a differentiable similarity function which is suitable for applying a gradient-based optimization method [22].

The mean-shift tracking algorithm is computationally fast, but it relies on small displacements in target position. This assumption can be thought as a drawback of the algorithm. Another drawback is that a mean-shift tracker cannot adjust to large appearance (scale, shape and color, etc.) changes of the target region during the tracking period [10]. Some scale adaptation schemes are applied in the literature [23, 24]. The cam-shift (continuously adaptive mean-shift) tracker in [24] is one of the successful methods, which uses the moment information. Finally, the mean-shift method can only handle infinitesimal partial occlusions.

3.1.2. Feature-Based Tracking

Instead of tracking the entire region, feature-based tracking algorithms track some extracted features of moving objects. Features can be the line segments, curve segments, corner vertices, etc. or a variety of distances and geometric relations between extracted features [3]. Distinguishable object features greatly simplify the tracking problem, because they are to be matched in successive frames.

The tracking technique proposed by [25] extracted corner features from every frame of a sequence. Figure 3.1 shows the corner features extracted for the selected frames of a toy car sequence. The extracted corners are used as measurements for the tracking filter. They demonstrate the ability of various tracking filters (like Kalman filters) using a variety of image sequences. The combination of Bayesian Multiple Hypothesis Tracking (MHT) technique and IMM algorithm handles the motion transition of features efficiently.

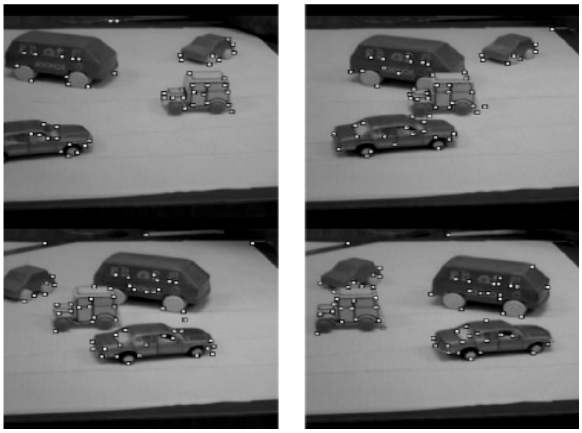


Figure 3.1. Corner features of toy cars

3.1.3. Boundary-Based Tracking

Apart from feature-based and region-based approaches, there are also boundary-based approaches. The region-based tracking algorithms take advantage of the entire region whereas boundary-based algorithms rely on the information provided by the object boundaries as in [26]. They used boundary-based information to detect and track several non-rigid moving objects over a sequence of frames. The boundary-based approaches like active contours extract the shape of the moving objects [12].

Active-contour-based tracking algorithms represent the outline (boundary) of a moving object as an evolving active contour, which is updated dynamically in successive frames [3]. Tracking can be performed by using state-space models to model the contour shape and motion or minimizing the contour energy by techniques such as gradient descent [14]. For example, [27] defined the object state in terms of spline shape parameters and affine motion parameters of the contour and updated them at each time instant using a particle filter where the measurements are the image edges computed in the normal direction to the contour [14]. Figure 3.2 is an illustration of a contour tracking result.

Figure 3.2. Tracking a hand across a desk

The active contour-based algorithms require an accurate contour initialization for each moving object. This means, moving objects have to be well separated (not occluded) during the initialization period.

4. DATA VALIDATION AND DATA ASSOCIATION

The foreground regions obtained from the object detection process are evaluated in the data validation and association step. For each tracked object, the data validation step ensures that the detected region in the current frame is originated from the tracked target by means of validation gating. When there is more than one validated region for the tracked object, data association algorithm selects the most relevant one.

4.1. Data Validation

Before providing the details of the data validation step, the Mahalanobis distance and its properties are described in detail. The Mahalanobis distance is essential in understanding the validation algorithm.

4.1.1. Mahalanobis Distance

Suppose an n-dimensional Gaussian random vector X with mean μ and covariance Σ. The locus (the set of points which satisfy a certain condition) for which the probability density function is greater than or equal to a specified constant can be defined as,

$$N(X;\mu,\Sigma)=\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left[-\frac{1}{2}[X-\mu]^T\Sigma^{-1}[X-\mu]\right]\geq K_p \tag{4.1}$$

This equation can be rewritten by defining $K=-2\ln\left((2\pi)^{n/2}K_p|\Sigma|^{1/2}\right)$ and then Equation 4.1 becomes

$$N(X;\mu,\Sigma)=[X-\mu]^T\Sigma^{-1}[X-\mu]\leq K \tag{4.2}$$

If there is equality in Equation 4.2 rather than an inequality, then

$$N(X;\mu,\Sigma) = \left[X-\mu\right]^T \Sigma^{-1}\left[X-\mu\right] = K \tag{4.3}$$

This equation is known as the squared Mahalanobis distance of vector X to the mean μ. The Mahalanobis distance is a normalized distance where normalization is achieved through the covariance matrix. In the special case where the covariance matrix Σ is a diagonal matrix with all its diagonal elements equal, then the Mahalanobis distance becomes equivalent to the Euclidean distance.

The locus of Equation 4.3 may be interpreted as the set of points of equal probability density represented by an n-dimensional ellipsoid centered about the mean μ. The ellipsoid semi-axes are the $\sqrt{K}$ times the square roots of eigenvalues of the covariance matrix Σ [7]. To illustrate this concept, if we assume a two-dimensional Gaussian vector $X = \left[x_1 \quad x_2\right]^T$ with mean $\mu = \left[\mu_1 \quad \mu_2\right]^T$ and diagonal covariance matrix,

$$\Sigma = \begin{bmatrix} \sigma^2{}_1 & 0 \\ 0 & \sigma^2{}_2 \end{bmatrix} \tag{4.4}$$

Then, the Equation 4.3 can be written for this particular case as

$$N(X;\mu,\Sigma) = \frac{(x_1-\mu_1)^2}{\sigma^2{}_1} + \frac{(x_2-\mu_2)^2}{\sigma^2{}_2} = K \tag{4.5}$$

The eigenvalues of the covariance matrix Σ are variance values $\sigma^2{}_1$ and $\sigma_2{}^2$, respectively. So, ellipsoid axis lengths are $2\sigma_1\sqrt{K}$ and $2\sigma_2\sqrt{K}$.

Therefore, the locus in Equation 4.2 is the border and inner points of the ellipsoid. The probability that a given value of the random vector X lies within the ellipsoid increases with the increase of K.

The scalar random variable K can be shown to be the sum of the squares of n independent zero-mean, unity variance Gaussian random variables, such a random variable has a chi-square distribution χ^2 with n-degrees of freedom [7].

When a particular probability value is desired, the value of K that yields an ellipsoidal region satisfying that probability can be obtained from chi-square probability density function. This probability is expressed as $100(1-\alpha)\%$ confidence level where α is the significance level.

$$\Pr\left\{K \le {\chi_\alpha}^2\right\} = \Pr\left\{[x-\mu]^T \Sigma^{-1}[x-\mu] \le {\chi_\alpha}^2\right\} = 1-\alpha \tag{4.6}$$

As an example, the 95% confidence level for two-dimensional Gaussian vectors can be obtained by the corresponding χ^2 distribution value for two-degrees of freedom and significance level $\alpha = 0.05$ which is ${\chi_{0.05}}^2 = 5.99$. Then, the ellipsoidal region can be defined as

$$\Pr\left\{K \le {\chi_{0.05}}^2\right\} = \Pr\left\{[x-\mu]^T \Sigma^{-1}[x-\mu] \le 5.99\right\} = 0.95 \tag{4.7}$$

4.1.2. Validation Gating

Validation gating establishes the confidence region for each tracked target where the occurrence of the target is expected to happen with a certain probability. In this section, the terminology and equations are held consistent with the filter equations and they are clarified in the relevant sections of Chapter 5.

Detected moving object region centroid position is the measurement information for the filter dedicated to the target. But, only one valid measurement is needed for each target. Before obtaining the true measurement $z(k)$ at the time labeled by k, the filter predicts the expected value of measurement $\hat{z}_j(k \mid k-1)$ for tracked target j in the filter prediction

step (Equation 6.38). The normalized squared Mahalanobis distance between each detected measurement $z_i(k)$ and predicted measurement $\hat{z}_j(k \mid k-1)$ is computed as

$$\upsilon_{ij}(k) = z_i(k) - \hat{z}_j(k \mid k-1) \tag{4.8}$$

$$S(k) = E\left[\upsilon_{ij}(k)\upsilon_{ij}(k)^T\right] = HP(k \mid k-1)H^T + R \tag{4.9}$$

$$d_{mah}{}^2 = \upsilon_{ij}(k)^T S(k)^{-1} \upsilon_{ij}(k) \tag{4.10}$$

The term $\upsilon(k)$ in Equation 4.8 is referred as innovation and $S(k)$ in Equation 4.9 represents the covariance of the innovation. Innovation covariance matrix can be thought as a measure of accuracy of the filter in predicting the target position. According to Kalman filter theory, it is shown in [7] that the innovation sequence is a zero mean Gaussian sequence and normalized innovation is chi-square distributed with the $\dim(z(k))$ (dimension of the measurement vector) degrees of freedom.

In relation to Mahalanobis distance, the validation gate can be defined as an ellipsoidal region centered at the predicted measurement $\hat{z}_j(k \mid k-1)$. This region can also be defined as a rectangular region but the ellipsoidal region is ideal for linear-Gaussian systems like the Kalman filter [28]. An ideal gate would have minimal volume in the measurement space for a given coverage probability [28]. The confidence region is defined as the inside of the ellipsoidal validation gate with gate threshold γ [7].

$$gate_{ij}(k) = \upsilon_{ij}(k)^T S(k)^{-1} \upsilon_{ij}(k) \le \gamma \tag{4.11}$$

The desired probability, $100(1-\alpha)\%$ confidence level, of the true measurement falling into the gate is determined by the gate threshold which can be obtained from the chi-squared distribution tables or direct calculations [6]. Increasing the desired confidence level will enlarge the confidence region which can be defined as,

$$\Pr\left\{\left[\upsilon_{ij}(k)^T S(k)^{-1}\upsilon_{ij}(k)\right] \le \chi_\alpha{}^2\right\} = 1-\alpha \tag{4.12}$$

So, the validity of a measurement is determined from the squared Mahalanobis distance of less than or equal to a gate threshold γ from the predicted measurement. The measurement $z_i(k)$ which falls inside the gate volume is assumed to be valid for track j and used in the filter estimation step to estimate the true target location in the image plane.

4.2. Data Association

When there are more than one validated measurements for one tracked object, data association algorithm selects the most relevant one. In this thesis, the Nearest Neighbor (NN) data association method is used. The validated measurement that is nearest to the predicted measurement (position) is selected and the rest is discarded. 'Nearest' means the minimum normalized squared Mahalanobis distance. This is illustrated in Figure 4.1 [29],

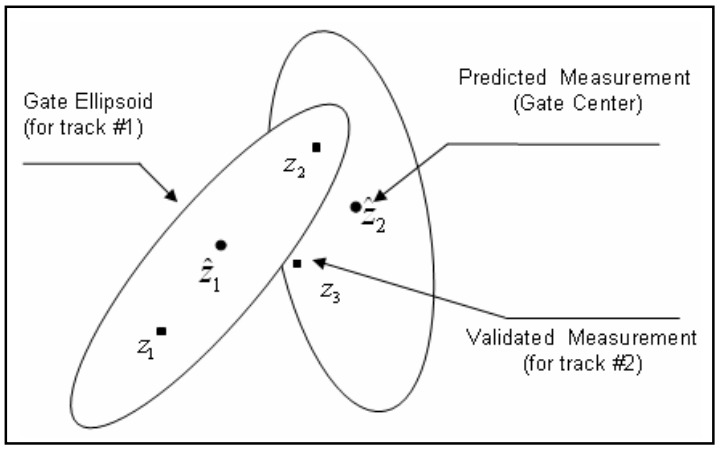


Figure 4.1. A data validation and association illustration

In this figure, there are two targets with predicted positions $\hat{z}_1$ and $\hat{z}_2$ and their validation regions, respectively. The measurements z_1 and z_2 are validated for the first target and the measurements z_2 and z_3 are validated for the second. The measurement z_2 is validated with both tracks but it is more close to the second target (track #2). Accordingly, the measurements z_1 and z_2 are associated to the first and second targets

respectively. Although the measurement z_3 is more close to $\hat{z}_1$ than measurement z_1, it is not in the validation region for the first target (track #1).

The main reason in using the NN method is that, it is easy to implement, computationally efficient, and adequate for our implementation. NN algorithm implicitly assumes loosely spaced targets and low rate of clutter in the gate. In this context, clutter refers to falsely detected spurious objects. When the targets are closely spaced, probabilistic data association techniques like Joint Probabilistic Data Association (JPDA) can be used. Instead of choosing only one validated measurement for each target and discarding the others, Probabilistic Data Association (PDA) techniques use all of the validated measurements with different weights (probabilities) [30].

5. STATE ESTIMATION

In the theory of stochastic processes, filtering is the estimation of the state of a dynamic (time varying) system from indirectly observed, noisy measurements [31]. Accordingly, the word filter is used with the meaning of reducing the effect of noise for obtaining the best estimate of the system state.

In this thesis, for each target, the measurement is the detected and validated moving object region centroid position in the image plane. The state variables are the subset of some particular properties of this centroid like position, velocity, and acceleration. The centroid position is a noisy observation due to the imperfections of the object detection step. Since targets are continuously affected by the problems mentioned in Section 2.1, the imperfections are inevitable. Tracking is performed by estimating the state of the target centroid in consecutive frames by using Kalman and IMM filtering techniques. The state estimation should localize the target as correct as possible in the current image plane and predict its future position in the next image plane.

Kalman filter is an optimum Minimum Mean Square Error (MMSE) estimator for linear dynamical systems. The IMM is a bank of Kalman filters running in parallel to increase the estimation accuracy. This chapter presents background information for Kalman filters.

5.1. State Estimation History

The history of optimal estimation theory starts with the Wiener filter [32]. Wiener filter is a spectral domain solution to the problem of reducing the amount of noise present in a signal by making a comparison with MMSE estimate of the desired noiseless signal. Both the signal and noise are assumed to be stationary linear stochastic processes with known spectral characteristics [33].

In the 1960's, Kalman [8] described the recursive solution to the optimal estimation problem of a random process using time-domain formulations with state-space methods. Recursive means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state. In contrast to the Wiener filter, the history of measurements and/or estimates is not required. This property makes the Kalman filter an effective state estimator. Kalman's original derivation uses orthogonal projections theorem. In the years that followed, simpler derivations based on statistical concepts are appeared.

At first, Kalman filter was used in trajectory estimation for the Apollo space program. Later on, many variations and extensions of Kalman filters have been developed. They find applications in several areas including aerospace, land, marine navigation systems, guidance of cruise missiles, nuclear power plant instrumentation, demographic modeling, manufacturing, telecommunications, the detection of underground radioactivity, fuzzy logic and neural network training, computer vision, and multi-sensor fusion [34].

5.2. State Estimation Considerations

In general, estimators model the noise as Gaussian white noise process. The following section describes the reasons for using white noise processes. The next section gives some desirable properties of estimators.

5.2.1. White Noise

Whiteness implies that its value is not correlated in time [35]. Stated more simply, if its value in present time is known, this knowledge makes no sense in predicting what its value will be at any other time. This means, the autocorrelation function and power spectral density of a stationary zero-mean continuous-time white noise "process" $w(t)$ are respectively [7],

$$R(\tau) = E[w(t)w(t+\tau)] = S_0\delta(\tau) \tag{5.1}$$

$$S(w) = F\{R(\tau)\} = \int_{-\infty}^{+\infty} e^{-jwt} R(\tau) d\tau = S_0 \tag{5.2}$$

In Equation 5.2, F refers to Fourier transform, w is the frequency and $\delta(\tau)$ is the continuous Dirac delta function (impulse function),

$$\begin{aligned} &\delta(t) = 0, \quad t \neq 0 \\ &\int_{-\varepsilon/2}^{\varepsilon/2} \delta(\lambda) d\lambda = 1, \quad \text{for any real number } \varepsilon > 0 \end{aligned} \tag{5.3}$$

Likewise, the discrete-time white noise is a random "sequence" with an autocorrelation function being a Kronecker discrete delta function $\delta[n]$ (instead of Dirac continuous delta function as in Equation 5.1) with properties,

$$\delta[n] = \begin{cases} 0 & n \neq 0 \\ 1 & n = 0 \end{cases} \tag{5.4}$$

A white noise autocorrelation function has constant power at all frequencies in the frequency spectrum. It is completely uncorrelated with itself at any time except the present. This property of white noise is very useful for design and analysis purposes.

Physical systems have a certain frequency range to which they can respond to inputs. Out of this range, either the input has no effect, or the system severely attenuates the input. Thus, the real wideband noise can be replaced by white noise that, from the system's point of view, is identical but the mathematics involved is substantially simple. As an illustration, a typical system and noise power spectral densities (the amount of power content at a certain frequency) are plotted in Fig. 5.1. [35].

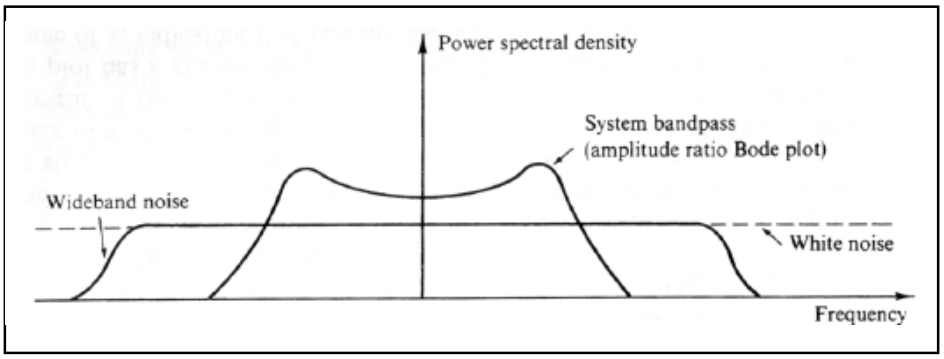


Figure 5.1. Power spectral density bandwidths

A system noise is typically emerges as a result of the cumulative effect of small sources. It can be shown mathematically by the central limit theorem that when a number of independent random variables are added together, the summed effect can be described very closely by a Gaussian probability density function, regardless of the shape of the individual densities [35]. Consequently, the Gaussian white noise provides mathematically simple models for various uncertainties inherent in the system.

5.2.2. Estimator Properties

Estimators have two naturally desirable properties, one of them is being unbiased and the other is having small Mean Squared Error (MSE). A state estimator bias represents the expected value of error in the state estimate and MSE provides a measure of the accuracy of the estimator [7].

Let x represent the state (usually a vector) we want to estimate, which is unknown, and let z represent observed/measured value of the state. Then, estimate $\hat{x}$ will be a function of the measurement z, denoted by $\hat{x}(z)$. The bias can be defined as,

$$bias\left[\hat{x}(z)\right] = E\left[\hat{x}(z)\right] - x = E\left[\hat{x}(z) - x\right] \tag{5.5}$$

A state estimator is said to be unbiased if its bias is equal to zero. Similarly, the variance and MSE of an estimator are respectively,

$$\text{var}\left[\hat{x}(z)\right] = E\left[(\hat{x}(z) - E\left[\hat{x}(z)\right])^2\right] \tag{5.6}$$

$$MSE\left[\hat{x}(z)\right] = E\left[(\hat{x}(z) - x)^2\right] = \text{var}\left[\hat{x}(z)\right] + (bias\left[\hat{x}(z)\right])^2 \tag{5.7}$$

If there is a bias in the estimate, this will increase the MSE which is the variance plus the bias squared. The next section describes the properties and design criteria of a Minimum Mean Square Error (MMSE) estimator.

5.3. Minimum Mean Square Error Estimator

A Minimum Mean Square Error estimator is defined as an estimator with minimal MSE which can be written mathematically as,

$$\hat{x}_{MMSE}(z) = \arg\min_{\hat{x}} E\left[(\hat{x}(z) - x)^2 \mid z\right] \tag{5.8}$$

The estimate which satisfies the above equation for vector random variables can be obtained by setting the gradient of the mean of the squared norm of this error to zero [7],

$$\nabla_{\hat{x}} E\left[(\hat{x} - x)(\hat{x} - x)^T \mid z\right] = 2(\hat{x} - E\left[x \mid z\right]) = 0 \tag{5.9}$$

$$\hat{x}_{MMSE}(z) = E\left[x \mid z\right] \tag{5.10}$$

Equation 5.10 uniquely defines the MMSE estimate as the conditional expectation (mean) of the state x given the measurement z. The MMSE state estimator for stochastic linear dynamic systems makes use of the initial state estimate and covariance with process and measurement models. They can be describes as follows,

Initial state estimate and covariance: The state is a random variable. According to Equation 5.10, the "mean" value of this random variable is initially set as our initial estimate of the state. Likewise, the "covariance" is initially set as a measure of the

accuracy of our initial estimate [7]. The estimator determines the future values of these distribution values.

Process (state evolution) model: The evolution of the state is modeled as a dynamic system perturbed by the process noise. This noise is used for modeling the uncertainties in dynamic models. The state-space representation for continuous-time LTI stochastic systems can be written as,

$$\dot{x}(t) = Fx(t) + Lw(t) \tag{5.11}$$

In this equation $x(t)$ represents the state vector, F is the system matrix describing system dynamics. L is the noise gain and $w(t)$ is the process noise. The input control term is discarded in Equation 5.11 since we don't have any control over the target state in our application.

Measurement model: Measurement model describes the dynamic model from the observers' perspective. Since measurements are obtained at discrete time instants, the discrete-time measurement equation can be written as,

$$z(k) = H_k x(k) + r(k) \tag{5.12}$$

In this equation, $z(k) \in R^m$ represents our measurement at time instant k. The $m \times n$ measurement matrix H_k establishes the "linear" relationship between the state $x(k) \in R^n$ and the measurement $z(k)$ in the absence of measurement noise $r(k)$.

Noise: In Equations 5.11 and 5.12, the noise terms model the disturbances which we can neither be controlled nor modeled deterministically. Accordingly, they are modeled as random variables having a certain distribution of possible values. In general, the process noise $w(t)$ is assumed to be stationary zero-mean white noise process with power spectral

density Q_c. Similarly, the measurement noise $r(k)$ is assumed to be stationary zero-mean white noise sequence with the covariance matrix R_k.

6. DISCRETE-TIME KALMAN FILTER

The discrete-time Kalman filter is the optimum MMSE estimator for the discrete-time linear dynamic systems under some statistical assumptions [7]. This section describes these assumptions and the realization of these assumptions in the proposed system. The algorithm is given with the computational origins of the filter at the end of the chapter.

6.1. Assumptions of the Discrete Kalman Filter

The assumptions of the discrete Kalman filter are related to process model and statistics of the state and model noise distributions.

6.1.1. Assumptions on Process Model

For the discrete Kalman filter, measurements occur and the states are estimated at discrete points in time. The discrete measurement model of MMSE estimator, as defined in Equation 5.12, can be used in discrete Kalman filter equations. However, the process model described in Equation 5.11 must be discretized and represented in the state-space form of discrete-time linear stochastic systems as

$$x(k+1) = A_k x(k) + q_k \tag{6.1}$$

The index $k+1$ represents the next sampling time after k. The $n \times n$ state transition matrix A_k defines how the state would change "ideally", in the absence of process noise sequence q_{k-1}, from the previous time step to the current time step. We want to emphasize the linear transition of the state variable from one time step to another by means of A_k matrix.

6.1.2. Assumptions on Initial State Estimate and Covariance

The state is modeled with a Gaussian distribution which can be completely described by the mean and covariance values. Since the Kalman filter is a MMSE estimator, according to Equation 5.10, the state estimate is the conditional expectation of the state given the measurements.

The Kalman filter needs initial "mean" and "covariance" values of the state. As in the MMSE, the initial "mean" value of the state is initially set as our initial estimate of the state,

$$E[x(0) \mid z(0)] = \hat{x}(0 \mid 0) \tag{6.2}$$

When this mean value is used in the covariance computation, then initial state covariance becomes,

$$\text{cov}[x(0) \mid z(0)] = E[(x(0) - \hat{x}(0 \mid 0))(x(0) - \hat{x}(0 \mid 0))^T \mid z(0)] = P(0 \mid 0) \tag{6.3}$$

The state covariance in Equation 6.3 is at the same time the covariance of the estimation error hence mostly referred as the "estimate error covariance" matrix. It represents the uncertainty in our initial estimate of $\hat{x}(0 \mid 0)$. After these initializations, the filter computes the future state estimations and uncertainties.

6.1.3. Assumptions on Process and Measurement Model Noise Sequences:

Considering the evolution of the system state as in Equation 6.1 and the measurement model in Equation 5.12, the process and measurement noise sequences are zero-mean Gaussian white noise sequences. Under this assumption, each noise sequence is represented by only a covariance matrix as

$$p(q_k) \sim N(0, Q_k) \tag{6.4}$$

$$p(r_k) \sim N(0, R_k) \tag{6.5}$$

The initial state, the process and measurement noises are mutually uncorrelated as

$$E[q_i q_j^T] = Q_i \delta[i-j] \qquad \forall i, j \in \mathrm{Z}^+ \tag{6.6}$$

$$E[r_i r_j^T] = R_i \delta[i-j] \qquad \forall i, j \in \mathrm{Z}^+ \tag{6.7}$$

$$E[q_k r_l^T] = 0 \qquad \forall k, l \in \mathrm{Z}^+ \tag{6.8}$$

$$E[x_0 q_k] = 0 \quad E[x_0 r_l^T] = 0 \qquad \forall k, l \in \mathrm{Z}^+ \tag{6.9}$$

6.2. Realization of Assumptions

In this section, how these assumptions are realized in our tracking system is described in detail.

6.2.1. Process Models

In filtering applications, the most important part is the proper selection of the process model. Because, incorrect modeling of the target's movements leads to higher inaccuracies in position estimation or may even result in divergence of the estimated target position from the actual position [36]. In this thesis, we use two target motion models; one of them is the nearly constant velocity model and the other is the turn model.

Small observation intervals and/or small target accelerations can be reasonably modeled by nearly constant velocity motion models like Continuous White Noise

Acceleration (CWNA) model [37]. We first describe the CWNA model in detail. In order to represent our process model in discrete-time as given in Equation 6.1, we discretized the CWNA model. Later on, we describe the turn motion model which greatly simplifies the estimation problem for the turning targets like parking cars.

The CWNA motion model: Consider again the stochastic differential equation of Equation 5.11 (repeated below),

$$\dot{x}(t) = Fx(t) + Lw(t)$$

After examining the image sequences we conclude that, most of the times humans and vehicles are moving at nearly constant velocities. The state vector containing the pixel coordinates of the target centroid and centroid velocities are adequate to describe these target movements at any time. Therefore, the state of the target can be represented by the four-dimensional column vector as

$$X = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T \tag{6.10}$$

Since movements are assumed at "nearly" constant velocities, the velocity component can change slightly during the course [7]. This change leads to an uncertainty in the target trajectory. Hence this unsteady acceleration, random velocity changes, can be modeled as a white process noise $w(t)$ giving the model name CWNA model [6] as

$$\ddot{x}(t) = w_x(t) \tag{6.11}$$

$$\ddot{y}(t) = w_y(t) \tag{6.12}$$

According to this model, the properties of the zero-mean Gaussian white noise process with a power spectral density Q_c and the associated process model matrices are,

$$E[w(t)] = 0 \tag{6.13}$$

$$E[w(t)w(t+\tau)] = Q_c\delta(\tau) \tag{6.14}$$

$$F = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.15}$$

$$L = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.16}$$

$$Q_c = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix} \tag{6.17}$$

The Q_c matrix in Equation 6.17 has the same uncertainty q in each Cartesian coordinate (diagonal entries) and these uncertainties are not mutually correlated (off-diagonal entries).

The uncertainty is determined according to the target's conformity to the selected process (dynamic) model and/or the noise present in the states itself. If the target doesn't deviate too much from its process model then Q_c should be set as small as possible $(q \sim 0.1)$ [25]. Relatively, large values $(q \sim 10)$ makes the model to act as a "nearly" acceleration model. Then, the filter quickly adapts to the measurements and makes noisy estimates for the position and velocity. These effects are further investigated in the filter algorithm part.

The discretized CWNA motion model: Since the CWNA model dynamics are LTI the discretization depends only on the time difference $\Delta t_k = t_{k+1} - t_k$. The discretized

matrices A_k and Q_k (the covariance of the discrete-time process noise sequence q_k) are as given below. For their derivation, please see [38]

$$A_k = \exp(F\Delta t_k) \tag{6.18}$$

$$Q_k = \int_0^{\Delta t_k} \exp(F(\Delta t_k - \tau))LQ_cL^T \exp(F(\Delta t_k - \tau))^T d\tau \tag{6.19}$$

In some cases Q_k can be calculated analytically. Even when a closed form solution is unavailable, the matrix can still be calculated efficiently using the following matrix fraction decomposition

$$\begin{pmatrix} C_k \\ D_k \end{pmatrix} = \exp\left\{ \begin{pmatrix} F & LQ_cL^T \\ 0 & -F^T \end{pmatrix} \Delta t_k \right\} \begin{pmatrix} 0 \\ I \end{pmatrix} \tag{6.20}$$

$$Q_k = C_k D_k^{-1} \tag{6.21}$$

Assuming A_k and Q_k to be constant, the resulting discretized matrices are computed as

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.22}$$

$$Q = \begin{bmatrix} \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & 0 & \Delta t & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 & \Delta t \end{bmatrix} q \tag{6.23}$$

In the remaining of the thesis state transition, measurement, process noise and measurement noise matrices are assumed to be constant and replaced by A, H, Q and R matrices respectively. This simplification is done for computational constrains.

The turn motion model: The turn motion model is similar to the discretized CWNA model. The process noise matrix is the same as Equation 6.23. The only difference is the state transition matrix which is,

$$A = \begin{bmatrix} 1 & 0 & \frac{\sin \omega \Delta t}{\omega} & \frac{\cos \omega \Delta t - 1}{\omega} \\ 0 & 1 & \frac{1 - \cos \omega \Delta t}{\omega} & \frac{\sin \omega \Delta t}{\omega} \\ 0 & 0 & \cos \omega \Delta t & -\sin \omega \Delta t \\ 0 & 0 & \sin \omega \Delta t & \cos \omega \Delta t \end{bmatrix} \tag{6.24}$$

where, the ω term represents the constant turn rate [25].

6.2.2. Measurement Model

In this thesis, the measurements are the outcomes of the object detection module which are the detected target positions in the image plane. An approximate measure of position errors (uncertainties) are represented by the measurement noise covariance matrix. The measurement matrix and measurement noise covariance matrix in Equation 5.12 (repeated below) are determined as

$$z(k) = Hx(k) + r(k)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.25}$$

$$R = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \tag{6.26}$$

The R matrix in Equation 6.26 has the same error variance σ^2 in each Cartesian coordinate (diagonal entries) and these errors are not mutually correlated (off-diagonal entries). The position errors are the cumulative effect of several factors like camera noise, partial occlusions, illumination changes in the object detection step. As σ^2 increases, the filter becomes slower to adapt the measurements and the estimates start to alter slowly. The effect of measurement noise is also investigated in the filter algorithm part.

6.2.3. Initial State and Initial State Covariance Estimates

In addition to the noise covariance values, the Kalman filter needs initial state estimate $\hat{x}(0|0)$ and initial estimate error covariance $P(0|0)$ values in order to start tracking. In our application, the estimate is initiated with the pixel coordinates of the first measurement with zero velocities in each coordinate. A few frames later, the filter will learn the velocity components. It seems reasonable to give the initial error covariance a small value. However, if $P(0|0)$ is too small the filter tightly bounds to the initial estimate and it takes a long time to adapt the measurements and to stabilize. The reason is that, we do not provide any prior knowledge about the target velocity. A good choice is to start with a high value of $P(0|0)$, then it will quickly decrease and converge to some nearly constant value [39].

6.3. The Discrete Kalman Filter Algorithm

6.3.1. The Computational Origins

This section gives an overview of the computational origins of the Kalman filter. The material presented in this section is mostly inspired from [39].

The A Priori and A Posteriori Error Covariance Matrix: At each iteration of the algorithm, the Kalman filter makes a prediction about the state $x(k)$. The prediction is made before the measurement information at time step k. This state is denoted by $\hat{x}(k \mid k-1)$ which is known as "a priori" estimate, i.e. "predicted" state. Then, the Kalman filter evaluates both the a priori estimate and the measurement information to make an estimate of the true value of the state $x(k)$. This state is denoted by $\hat{x}(k \mid k)$ and known as "a posteriori" estimate, i.e. "estimated" state. The corresponding a priori and a posteriori estimate errors with respect to the true value of the system state can be defined as

$$e_k^{-} = x(k) - \hat{x}(k \mid k-1) \tag{6.27}$$

$$e_k = x(k) - \hat{x}(k \mid k) \tag{6.28}$$

Then, the a priori and a posteriori error covariance matrices associated with each estimate error can be given as,

$$P(k \mid k-1) = E\left[e_k^{-} e_k^{-T}\right] \tag{6.29}$$

$$P(k \mid k) = E\left[e_k e_k^{T}\right] \tag{6.30}$$

A posteriori error covariance matrix represents of the accuracy of the state estimate.

The Optimal Kalman Gain: In Kalman filtering, the goal is to compute the a posteriori state estimate $\hat{x}(k \mid k)$ as a linear combination of the a priori estimate $\hat{x}(k \mid k-1)$ and a weighted difference between the actual measurement $z(k)$ and the predicted measurement $H\hat{x}(k \mid k-1)$ as

$$\hat{x}(k \mid k) = \hat{x}(k \mid k-1) + K(k)\left[z(k) - H\hat{x}(k \mid k-1)\right] \tag{6.31}$$

The difference $z(k) - H\hat{x}(k \mid k-1)$ is the so-called the measurement "innovation" or the "residual". The residual reflects the discrepancy between the predicted and the actual measurements. The matrix K in Equation 6.31 is the "gain" or blending factor. The gain term simply makes a correction on the a priori estimate $\hat{x}(k \mid k-1)$ in order to yield the a posteriori state estimate $\hat{x}(k \mid k)$.

The trace of a square matrix is defined to be the sum of the elements on the main diagonal. Therefore, the trace of a posteriori error covariance matrix $P(k \mid k)$ gives the sum of the mean squared error. Since the Kalman filter is a "minimum" mean square error estimator, the trace of the $P(k \mid k)$ matrix must be minimized by setting the value of gain K to an appropriate value. This minimization can be accomplished by first substituting Equation 6.31 into the definition of a posteriori estimate error, $e(k)$, and substituting it into Equation 6.30. After performing the indicated expectations, the derivative of the trace of the result with respect to Kalman gain is equalized to zero. One popular form of the solution is given as

$$K(k) = \frac{P(k \mid k-1)H^T}{HP(k \mid k-1)H^T + R} \tag{6.32}$$

The Kalman gain determines the effect of obtained measurement and predicted value of the measurement on the state estimate according to the measurement and process noise covariances, R and Q respectively. The relation between measurement noise covariance R and the Kalman gain is clear. However, the effect of the process noise

covariance can be seen in the a priori error covariance $P(k|k-1)$ equation later on. Taking a simple view of the optimal filter gain in Equation 6.32 and a posteriori state estimate in Equation 6.31 we can conclude that,

i) When the measurement noise covariance R approaches zero (a trusted measurement), then a posteriori estimate is nearly equal to the measured value.

ii) When the a priori error covariance $P(k|k-1)$ approaches zero (a trusted process model), then a posteriori estimate is nearly equal to the a priori estimate.

iii) If the measurement noise covariance is relatively high with respect to process noise covariance (measurement is less accurate than process) the gain will be small and filter puts more confidence on the process model.

iv) Conversely, if the process noise covariance is relatively high the gain will be large and filter trusts measurements more.

6.3.2. The Algorithm

The Kalman filter recursively computes the MMSE estimates of the state of a dynamic system through the time and measurement update steps. Each iteration, except from the first one, starts with a posteriori estimates of the previous cycle. For the derivation of these equations, we refer to [38], [7].

Time Update: Starting from the initial state estimate $\hat{x}(0|0)$ and estimate error covariance matrix $P(0|0)$ as defined in Section 6.2.3; this step predicts their corresponding values for the next sampling time. This prediction is based on the state evolution model in Equation 6.1. The prediction equations are as

$$\hat{x}(k \mid k-1) = A\hat{x}(k-1 \mid k-1) \tag{6.33}$$

$$P(k \mid k-1) = AP(k-1 \mid k-1)A^T + Q \tag{6.34}$$

Measurement Update: When the new measurement is obtained, its value is used to correct/improve the predicted values coming from the Time Update step. The correction equations are as

$$K(k) = P(k \mid k-1)H^T \left[HP(k \mid k-1)H^T + R\right]^{-1} \tag{6.35}$$

$$\hat{x}(k \mid k) = \hat{x}(k \mid k-1) + K(k)\left[z(k) - H\hat{x}(k \mid k-1)\right] \tag{6.36}$$

$$P(k \mid k) = \left[I - K(k)H\right]P(k \mid k-1) \tag{6.37}$$

The predicted state of the target $\hat{x}(k \mid k-1)$ in Equation 6.33 is used to predict the position of the target for the next sampling time by multiplying it with the measurement matrix as

$$\hat{z}_j(k \mid k-1) = H\hat{x}(k \mid k-1) \tag{6.38}$$

This predicted measurement is used along with the innovation covariance matrix (the $HP(k \mid k-1)H^T + R$ term in Equations 6.35 and 4.11 in the data validation algorithm of Section 4.1.2. Then, the validated measurement is used in Equations 6.36 in order to make the state estimation.

The Equations 6.34 and 6.37 updates the accuracy statistics of the filter: when targets are under occlusion, the object detection module could not determine the accurate position of target centroids. Then, the measurement update step is skipped because there are no available measurements. The Kalman filter efficiently tolerates these situations by propagating the predictions forward in time. Nevertheless, the accuracy of the estimate

decreases with every iteration. As Equation 6.34 states, if the measurement step is skipped the uncertainty of estimate grows due to the process noise. However, if target deviates too much from the process model or occlusions last very long, the filter will operate wrongly.

Since the prediction (time update) step largely depends on the process model, the filter will obey this model until a measurement arrives. As the measurements arrive, the estimate error covariance value decreases (toward its lower limit) because the new information reduces the uncertainty (Equation 6.37).

6.4. Summary of Kalman Filters

To summarize, Kalman filter makes the best utilization of the prior knowledge of the system dynamics to produce an optimal estimate of the state in such a manner that a posteriori error covariance is minimized statistically when some presumed conditions are met. The Kalman filter updates only the state estimate and covariance matrices as they give sufficient statistics to summarize the entire past and make the best estimate.

Kalman filter is an efficient recursive estimator for linear dynamical systems discretized in time domain. Linear means that the process and measurement models are linear functions of the state variable. For nonlinear systems (process and/or measurement models) extended and unscented Kalman filters have been proposed in the literature.

There are also alternate but equivalent formulations of the Kalman filter equations. Some of them are sequential Kalman filter and information filter. They provide efficient implementation of the linear estimation techniques. Sequential Kalman filtering allows for the implementation of the Kalman filter without computing matrix inversions. Information filtering propagates the inverse of the covariance matrix P^{-1} [40]. Information filtering is computationally simpler than Kalman filtering under certain conditions.

7. INTERACTING MULTIPLE MODEL ESTIMATOR

The IMM estimator using Kalman filters is presented in this chapter. In the explanation of the Kalman filter, we used a single motion (process) model. However, sometimes more than one model is needed to represent the motion of targets during their course (i.e. a single constant velocity model is inadequate to model the maneuvers like turns and accelerations). In such cases, multiple model algorithms are needed to use multiple models simultaneously.

Amongst the multiple model techniques, the IMM was shown to achieve an excellent compromise between performance and complexity [41]. It was originally proposed by [42] and successfully used in the implementation of an air traffic control system. The IMM was further developed by [7] and used in applications where multiple and different sensors along with highly-maneuverable targets are present [41]. The following sections present a detailed description of the IMM algorithm and some design considerations.

7.1. Algorithm

The IMM estimator uses a bank of filters each based on a different state-space model which corresponds to a particular mode of target movements like turning, accelerating, and not accelerating. In this thesis, we used Kalman filters as building blocks of the IMM algorithm.

The IMM is a recursive algorithm. Figure 7.1 describes the overall IMM algorithm with two interacting filters operating in parallel. The equations involved in one iteration cycle can be divided into four steps: calculation of the mixing probabilities, mixing (interacting) of the estimates, mode-matched filtering and mode probability update. The steps are performed in parallel for each Kalman filter. For the derivation of these equations, please see [7].

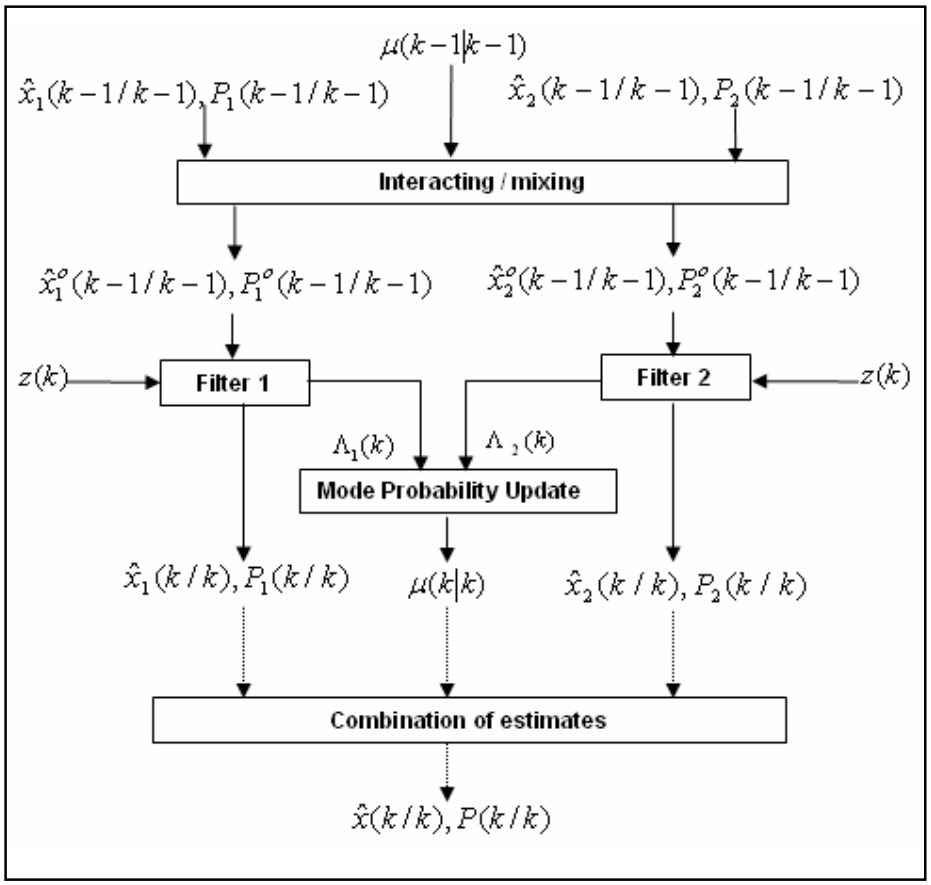


Figure 7.1. An IMM cycle.

At the end of an iteration, each filter computes its own state estimate and estimate error covariance, and also associated model likelihood. At the beginning of an iteration, each filter weights (suitably mixes/combines) these estimation results from all filters and aggregates them. Weights are the so-called mixing probabilities. This mixing/interaction makes the IMM estimator to maintain all of the filters in track [41]. As a result of this, it is capable of adapting itself to the more likely motion model (target maneuver) available from the bank of models.

i) Calculation of the mixing probabilities:

The mixing probability for mode j at time step k is calculated under the assumption that mode i was in effect at $k-1$ as

$$\mu_{i|j}(k-1\,|\,k-1)=\frac{1}{\bar{c}_j}\,p_{ij}\mu_i(k-1) \qquad i,j=1,....,r \tag{7.1}$$

In this equation, how probable it is that the system was in mode i is denoted by $\mu_i(k-1)$, called the (predicted) probability of mode i, and how probable it is that the system will switch to mode j from mode i is denoted by p_{ij}. Finally $\bar{c}_j$ is the normalization constant,

$$\bar{c}_j=\sum_{i=1}^{r} p_{ij}\mu_i(k-1) \qquad j=1,....,r \tag{7.2}$$

The "mode probabilities" $\mu_i(0)$ and "mode transition probabilities" p_{ij} are assigned by the user a priori but the latter doesn't change with time. The considerations for setting the mode transition probabilities are explained in Section 7.3.

ii) Mixing (Interacting) of the estimates:

The state estimate and estimate error covariance of the Kalman filter corresponding mode i are denoted as $\hat{x}^i(k-1\,|\,k-1)$ and $P^i(k-1\,|\,k-1)$ respectively.

Since we assume that mode j is the active mode at k, we want to update the filter equations corresponding to mode j with a new measurement $z(k)$. At this point, the filter cannot use directly the estimates from Kalman filter matched to mode i but instead it "mixes" the estimates (outputs) of all the filters (obtained in the previous time step) by mixing probabilities $\mu_{i|j}(k-1\,|\,k-1)$ as

$$\hat{x}^{0j}(k-1\,|\,k-1)=\sum_{i=1}^{r}\hat{x}^i(k-1\,|\,k-1)\mu_{i|j}(k-1\,|\,k-1) \qquad j=1,....,r \tag{7.3}$$

$$P^{0j}(k-1|k-1)=\sum_{i=1}^{r}\mu_{i|j}(k-1|k-1)\left\{P^{i}(k-1|k-1)+\left[\hat{x}^{i}(k-1|k-1)-\hat{x}^{0j}(k-1|k-1)\right]\cdot\left[\hat{x}^{i}(k-1|k-1)-\hat{x}^{0j}(k-1|k-1)\right]^{T}\right\} \quad (7.4)$$
$$j=1,....r$$

This way, the input to the filter matched to mode j is obtained from an interaction of the r filters.

iii) Mode-matched filtering:

The state estimate in Equation 7.3 and its covariance Equation 7.4 are used as inputs to the Kalman filter matched to mode j to obtain updated (a posteriori) state estimate $\hat{x}^{j}(k|k)$ and covariance $P^{j}(k|k)$ respectively with measurement $z(k)$. This Kalman filtering is done as described in Chapter 6.

As a measure of how likely it is that the model used in Kalman filter (matched to j) is the correct one, the likelihood function $\Lambda_j(k)$ of mode j is computed as

$$\Lambda_j(k)=N\left[z(k);\hat{z}_j(k|k-1),S_j(k)\right]=N\left[\upsilon_j(k);0,S_j(k)\right] \quad (7.5)$$
$$j=1,....,r$$

The $\upsilon_j(k)=z(k)-\hat{z}_j(k|k-1)$ term is residual and $S_j(k)$ is its covariance.

iv) Mode probability update:

The new mode probability $\mu_j(k)$ is computed using the mode likelihood function as

$$\mu_j(k) = \frac{1}{c}\Lambda_j(k)\bar{c}_j \qquad j = 1,....,r \tag{7.6}$$

The $\bar{c}_j$ term is as defined in Equation 7.2 and c is the normalization constant,

$$c = \sum_{j=1}^{r} \Lambda_j(k)\bar{c}_j \tag{7.7}$$

7.2. Combination of Estimates and Covariances

The overall state estimate and covariance of the IMM filter is the sum of the model-conditioned estimates and covariances of all filters weighted by the corresponding model probabilities as

$$\hat{x}(k \mid k) = \sum_{j=1}^{r} \hat{x}^j(k \mid k)\mu_j(k) \tag{7.8}$$

$$P(k \mid k) = \sum_{j=1}^{r} \mu_j(k)\left\{P^j(k \mid k) + \left[\hat{x}^j(k \mid k) - \hat{x}(k \mid k)\right]\cdot\left[\hat{x}^j(k \mid k) - \hat{x}(k \mid k)\right]^T\right\} \tag{7.9}$$

These combinations can be calculated at each iteration. However, they are not used in the algorithm cycle. These equations are for output purposes. We use only Equation 7.8 to represent the target position estimate in the image plane.

7.3. Determining Mode Transition Probabilities

A proper choice is to derive the diagonal coefficients of the transition probability matrix from the expected sojourn time τ_i (in units of the sampling interval) in each model i [7]. Sojourn time is the typical amount of time that we expect the target to stay in that mode [6]. The probability of transition from model i to itself p_{ii} is roughly,

$$p_{ii} = 1 - \frac{T}{\tau_i} \tag{7.10}$$

The T term is the sampling interval (i.e. time between two consecutive frames). The remaining probability $(1 - p_{ii})$ for model i can be shared equally between the off-diagonal entries as

$$p_{ij} = \frac{1 - p_{ii}}{m - 1} \qquad i \neq j \tag{7.11}$$

The performance of the IMM algorithm is not very sensitive to the choice of the transition probabilities including the off-diagonal elements. The only exception is that the matrices with diagonal entries less than 0.75 degrade the performance of the tracker [43].

7.4. Validation Gating For IMM Filtering

For the IMM filter, there is a different innovation and innovation covariance $S(k)$ (Equation 4.11) for each Kalman filter in the combined model. According to the Centralized Gating method, the validation gate center (predicted measurement) for r models can be found as a weighted combination of each model prediction with the probability associated with each model $\mu_m(k-1)$ as

$$\hat{z}(k \mid k-1) = \sum_{m=1}^{r} \hat{z}_m(k \mid k-1)\mu_m(k-1) = \sum_{m=1}^{r} H\hat{x}^m(k \mid k-1)\mu_m(k-1) \tag{7.12}$$

The model probability comes from the previous time step $k-1$ [44].

The matrix with the largest determinant $|S_r(k)|$ is chosen as the innovation covariance matrix. The other method can be the Model-Based Gating in which there is no single gate center and region, each model establishes its own gates. All validated measurements from all gates are taken into consideration for subsequent processing. In this thesis, Centralized Gating method is used for the data validation step.

8. EXPERIMENTAL RESULTS

This chapter presents the experimental results of the algorithm implemented on PETS (Performance Evaluation of Tracking and Surveillance) datasets. This dataset originates from Second IEEE International Workshop on PETS 2001. We run our algorithm on a PC having Intel Core 2 Quad Q9000 processor with 3 GBytes RAM in MATLAB. The first section briefly describes the test material used. The algorithm improvements and results are presented in the following sections. At the end of the chapter, median-based background subtraction and mean-shift tracking algorithms are implemented and the results are compared with the proposed method.

8.1. Test material description

PETS 2001 dataset provides a unique test environment for the objective evaluation of the tracking algorithms. Dataset scenarios take place in a car park environment. These scenarios contain the most problematic tracking situations like illumination changes, varying reflectance, shadows, and repetitive background motions such as moving clouds, small camera displacements, swaying branches, and leaves. In some scenarios, there are similar colored objects with the background making it difficult to distinguish between the two and stationary scene structures like street lamp or tree blocking the view of moving objects. The dataset consist of 576x768x3 pixel color image sequences taken at 25 fps.

8.2. Algorithm improvements

We improved on the standard Mixture of Gaussians object detection method for illumination changes and stopped objects. We also improved the data validation step to handle occlusion more reliably. Finally, we improved the track initialization step. Our aim in these was to have a more robust system. We also considered the real-time constraints. At first, in order to speed up the algorithm, we used one out of two images from the dataset. We decreased the resolution of the images to 173x231 pixels and converted them to monochrome (grayscale) intensities.

8.2.1. Adaptations on Mixture of Gaussians method

In dataset scenarios, there occur sudden illumination changes affecting the whole frame. Actually, the Mixture of Gaussians is an adaptive algorithm. But, high adaptation rates handle lightning changes at the cost of learning slowly moving objects as background. Therefore, we choose a small learning rate enough to handle slow illumination changes. To handle sudden illumination changes, the method is further made adaptive by subtracting the mean value of the incoming image with the average of the first Gaussian distribution mean values. If the difference is above a certain threshold, the mean value of the first Gaussian distribution in the mixture is shifted by an amount proportional to the difference and variance value is increased. After this adaptation, the algorithm becomes more attentive to the lightning changes. So, the brightness values of the incoming images are compared with the background image in each frame. If many pixels are affected by brightness, then the mean value check is performed again. This gradual adaptation scheme does not destroy or eliminate the detected objects. Thereby, slowly moving objects are not incorporated into the background and lightning changes are handled appropriately.

The other important adaptation is about stopped objects. In dataset scenarios, there are temporarily or totally stopped objects. If one of the tracked objects is in the vicinity of a stopped object, we incorporate the stopped object into the background model. Since there are two Gaussian distributions which model the background pixel values, we assign the stopped object pixel values to the one of the Gaussian distribution mean values. So, the stopped object is learned as background very quickly and do not disturb the detection results of other objects. If we incorporate every stopped object in the background model, we will loose the track of temporarily stopped objects.

8.2.2. Improvements on data validation and occlusion

The tracked objects are perceived as larger and faster as they move towards the camera and vice versa. However, the similar effects can be caused by occlusions. When a stationary scene structure is blocking the view of a moving object, the detected object size is smaller than the original size or the object could not detected at all. Also, when moving

objects come very close to each other, the detected region is bigger than individual object sizes. Therefore, we track the centroids of moving objects as they are not much affected with appearance changes.

The centroid motions are the most important cues for locating the objects in the image plane especially when they are under occlusion. The IMM filter dedicated to each target predicts the possible location for that target centroid in the current image plane. Validation gating establishes the confidence region around the predicted location where the occurrence of the target is expected to happen with 99% probability. The nearest detected object to our target whose centroid is falling in the gate is validated for further processing. When the predicted positions for different targets are very near as to introduce ambiguity, none of them validates a detected object.

The other consideration is, in order to ensure the right object to validate and to detect occlusions; the regional properties of the validated foreground object (blob) are compared with the tracked object. These properties are size, orientation and length of the major axis of the ellipse which has the same normalized second moment as the detected region. The confirmed region centroid is used in the filter equations to locate the tracked object and update the filter accuracy statistics.

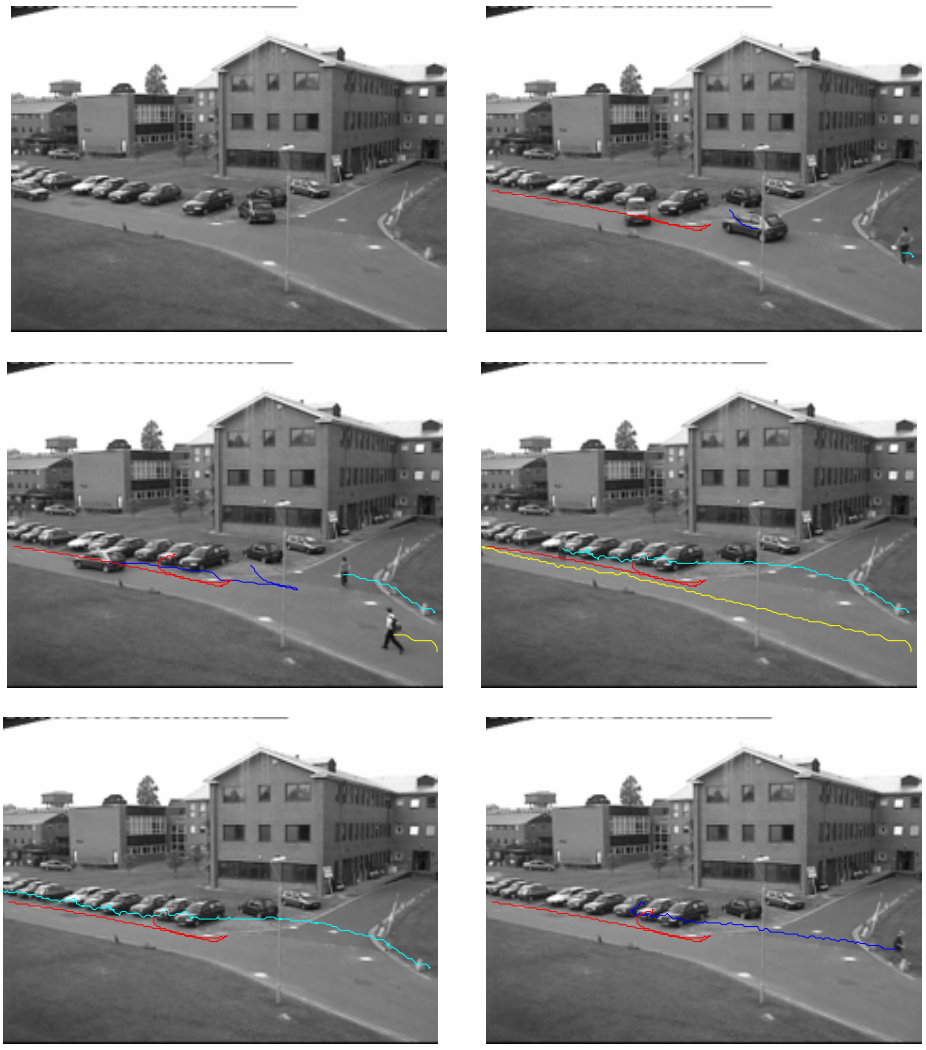
8.2.3. Improvements on track initialization

Detected but not validated objects can be initialized as new targets. The criterion is that, they have to be apart from existing targets. The most important reason is, as mentioned before, inter object occlusions form a single moving region in the foreground image and the tracked targets won't validate this merged region. But the region does not belong to a new object. Rather a few frames later the moving objects will move away from each other and the constituent parts will split up.

8.3. Test Results

We tested the algorithm on four different image sequences taken from two different camera views and settings. The color properties of each sequence are different from others. However, the object detection algorithm is run with the same parameter set for all of them. The results are illustrated with screenshots taken at critical points in each scenario. In each of the screenshots, the moving object locations are linked and superimposed throughout the length of the sequence and the trajectories are obtained.

i) Dataset 1 - Camera 1



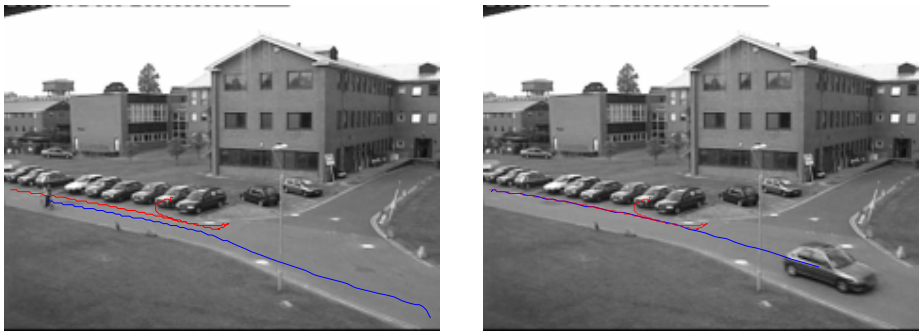


Figure 8.1. Frames 980, 1276, 1456, 1882, 2194, 2532, 2910, 3028

At the beginning of the first dataset, between frames 980 and 1360, the car with the blue trajectory is departing from the parking position very slowly and the other car with the red trajectory is parking to another position rapidly. Later, two persons start walking from the right side of the camera view to the left. When the person with the cyan trajectory comes very close to the newly parked car, at frame 1650, the algorithm learns the stopped car as a background object. So, both the pedestrian and the person leaving the car are detected properly. Although the pedestrians with yellow and cyan trajectories become very small in the image plane, they are tracked properly until they leave the scene. Starting from frame 2940, a car moves towards the camera with an increasing velocity.

In this scenario, the lamppost partially occludes the moving objects and splits the detected object regions into two parts. If one of the splitting parts is wrongly validated, the other part can be initiated as a new object. But our validation control and new track initialization algorithms are well-prepared for these situations.

ii) Dataset 1 - Camera 2

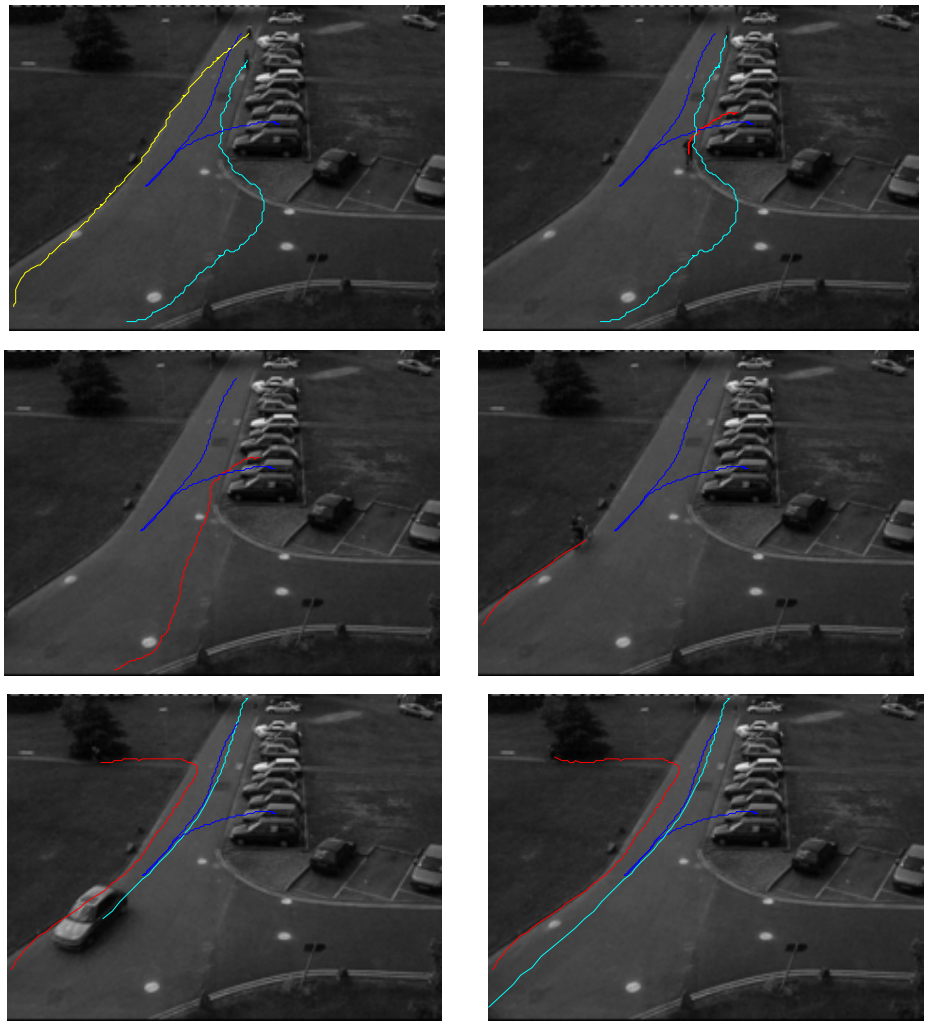
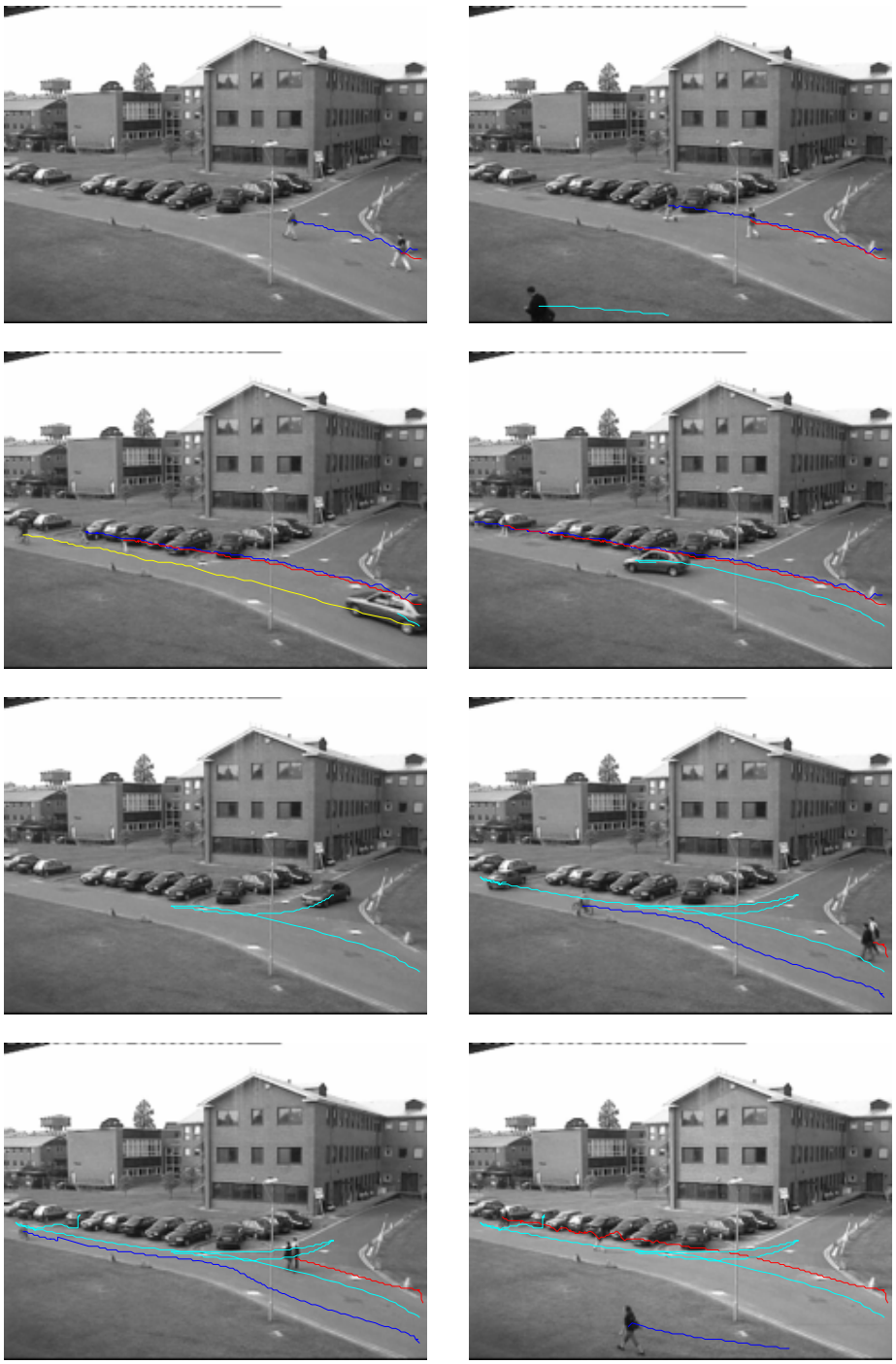


Figure 8.2. Frames 1000, 1536, 2100, 2342, 2766, 2812, 3016, 3036

The second camera sequences are very dark. The number of frames is nearly same but this view represents the scenario more clearly. For example, the cycling man changes his course when the car with the cyan trajectory comes close to it. This information is lost in the previous sequence.

iii) Dataset 2 - Camera 2



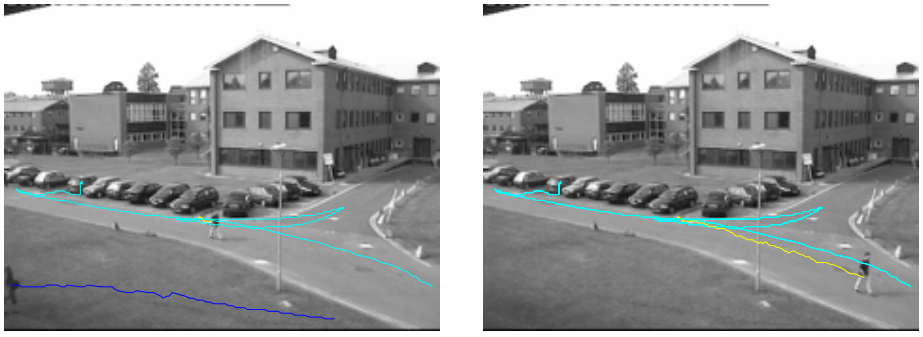
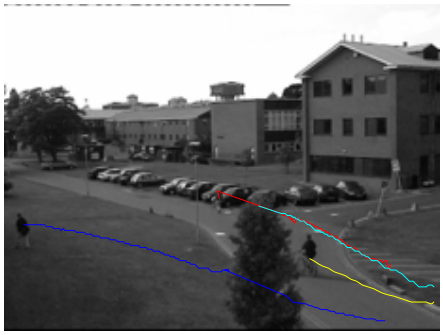
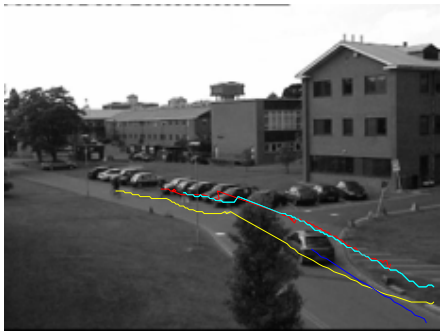
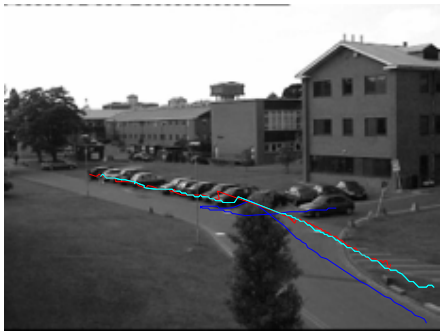


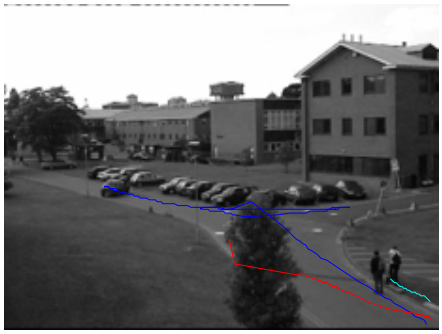
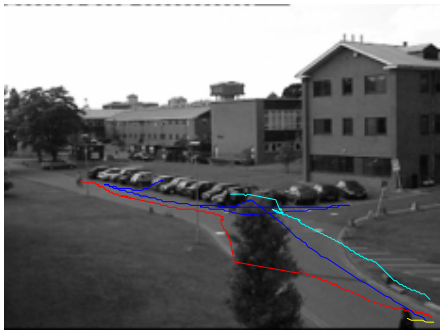
Figure 8.3. Frames 680, 776, 968, 1122, 1250, 1826, 1952, 2382, 2478, 2624

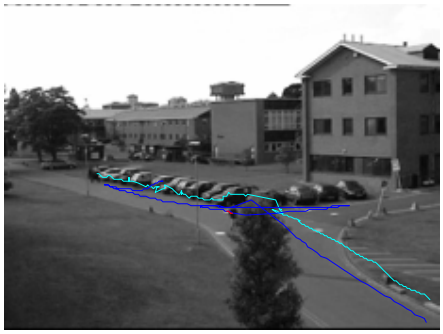
In this scenario, the pedestrians and cycling man are tracked until they leave the scene. Around frame number 960, there appears a highly maneuvering car. The IMM filter with velocity and turn models easily tracks the target.

The second dataset has significant illumination changes than the first one. When we compare the first and final snapshots, the lightning increase can be seen more easily. Our gradual adaptation scheme performs well. The illumination changed pixels are not falsely detected as moving objects. Besides, real moving objects are accurately found and tracked in the sequence. The only exception can be seen in frame 2478. The algorithm fails to initialize the track of the person leaving the parked car. Because, as he walks very near to the parking cars, he is very similar colored with the background. In this special case, the tracking is started when he departs from the parking area. Between frames 1820 and 2400 two pedestrians are walking so close in the image that they are extracted as one region.

iv) Dataset 2 - Camera 1



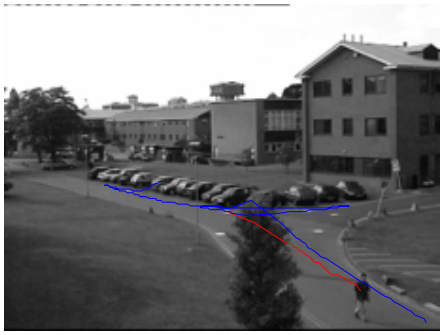


Figure 8.4. Frames 485, 601, 791, 961, 1219, 1545, 1675, 2069, 2479, 2535, 2779

In this scenario, the illumination changes are much more significant than the other scenarios. Also, the camera view is darker than the first one and there is a big tree blocking its view. All of the moving objects are fully occluded with the tree at least once. They appear after many frames, and are correctly validated by our system. The accurate filter predictions greatly reduced the validation problem. When objects are passing behind the tree, their predicted locations are also plotted.

We also implement the same scenario with a single Kalman filter. The motion model is the same velocity model of the IMM filter. The elapsed time is 1887 seconds for Kalman filter and 1918 seconds for the IMM filter. For uniformly moving targets, there is not much difference between the estimates of the IMM filter and the Kalman filter. However, during turns the IMM filter has much better speed and position estimates at the cost of 31 seconds.

8.4. Algorithm Evaluation

The Tracking Accuracy (TA) of our algorithm is evaluated with the manually generated ground truth data. Data represents the true trajectories of targets. We define True Positives (TP) as the number of observations located within the ground truth data and False Positives (FP) as the number of observations not located within the ground truth data. Finally, False Negatives (FN) can be defined as the number of frames which the tracking algorithm not observed any locations for the tracked objects. Then TA can be computed as

$$TA = \frac{TP}{TP + FP + FN} \tag{8.1}$$

The evaluation results can be seen in Table 1. As can be seen in this table, our method works fairly well on diverse test sets. Last row represents the evaluation results of the single Kalman filter.

Table 8.1. The evaluation results

	Scenario	TP	FP	FN	TA(%)
Proposed Method	Dataset 1-Camera 1	1849	46	21	96.5
Proposed Method	Dataset 1-Camera 2	1447	64	6	95.3
Proposed Method	Dataset 2-Camera 1	2093	93	97	91.6
Proposed Method	Dataset 2-Camera 2	1819	40	76	94.0
Kalman Method	Dataset 2-Camera 1	1883	164	236	82.4

8.5. Algorithm Comparisons

In this section, median-based background subtraction and mean-shift tracking algorithms are implemented for comparison purposes.

8.5.1. Median based background subtraction algorithm

The median based algorithm is simple to implement when compared to Mixture of Gaussians method. First, at each pixel location, the median value of the previous N frames is computed. Then, median values are subtracted from the incoming frame pixel values. The pixels where the difference is above a certain threshold are assigned as moving object pixels.

We used the past 20 frames in order to not degrade the speed of the algorithm too much. The performance of the algorithm is evaluated by running the Dataset 2-Camera 2 sequence on the same computer for two background subtraction methods. The object detection accuracies are nearly same. But, the elapsed time is 1918 seconds for Mixture of Gaussians method and 4832 seconds for the median-based method. However, the main

limitation is, it cannot be used for situations where the moving objects are temporarily stopped for more than N frames. In this case, the algorithm incorporates the stopped object in the background model. When the object starts to move again, nearly $N/2$ frames later it is recognized as a moving object.

8.5.2. Mean-shift tracking algorithm

The mean-shift tracking algorithm is implemented for tracking quality comparisons. We applied the algorithm on a simple and short frame sequence where a woman is walking along the road. Figure 8.5 illustrates the tracking result for the mean-shift tracker.

Figure 8.5. Frames 2050, 2290

Since the mean-shift tracker could not adjust to scale changes, the algorithm fails to track the woman as she is moving away from the camera and getting smaller. Our algorithm tracked the woman correctly until she leaves the camera view as illustrated in Figure 8.6.

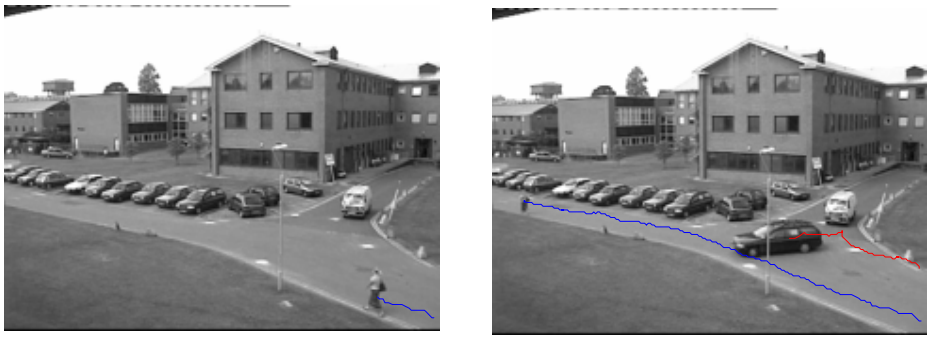


Figure 8.6. Frames 2050, 2500

9. CONCLUSION

The automated visual surveillance system presented in this thesis is designed to accurately cope with changing background conditions like illumination levels, shadows, and repetitive background motions such as moving clouds, small camera displacements, swaying branches and leaves. In addition, the system properly handles both temporary and totally stopped objects. These are accomplished by making some adaptations on Mixture of Gaussians object detection method.

The detected objects are validated for the tracked targets according to their distances to predicted target positions and the errors made in these predictions by validation gating and NN data association algorithms. However, to ensure the right object to validate; we further make some controls on the validated object region. The control criterions are size, orientation and length of the major axis of an ellipse which has the same normalized second moment as the tracked object region. Their limits are selected for normal object appearance changes according to usual movements and relative distances to camera. The validated object region centroid position is used in the IMM estimator in order to estimate the true location of the target. Since the IMM estimator operates on both constant velocity and turn motion models simultaneously, estimation results are far more robust than any single model based estimator. When the algorithm does not validate any object for the tracked target, the IMM estimator efficiently makes the predictions about the possible target locations in the image plane. The primary reason for non validation is the occlusions either by static scene structures or between moving objects.

This thesis mainly focused on developing a robust tracking algorithm with real-time processing capability. We track people and vehicles on four image sequences provided by PETS 2001 datasets and the overall tracking accuracy is computed as 94.3% according to the manually generated ground truth data. On a total of 8676 frames, the processing time is 0.8sec per frame.

In the future, the algorithm could be made more robust by discriminating between the closely spaced moving pedestrians from each other. The main limitation of the algorithm is that it not designed to cope with congested situations. Our system can be used in less congested public places like building complexes or ministry buildings.

REFERENCES

1. Kumar, P., A. Mittal and P. Kumar, *"Study of robust and intelligent surveillance in visible and multimodal framework",* Informatica, Vol. 32, pp. 63–77, April 2008.

2. Collins, R. T., A. J. Lipton and T. Kanade, *"Introduction to the special section on video surveillance", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 745-746, August 2000.

3. Hu, W., 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., August 2004.

4. Conte, P. D., *Detection, Tracking, and Behavior Analysis of Moving People in Intelligent Video Surveillance Systems: A Graph Based Approach*, Ph.D. Thesis, L'institut National des Sciences Appliquées de Lyon, 2006.

5. Stauffer, C. and W. E. L. Grimson, *"Adaptive background mixture models for real-time tracking", Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* Vol. 2, pp. 252, CO USA, June 1999.

6. Peters, D. J., *A Practical Guide to Level One Data Fusion Algorithms,* DREA Technical Memorandum No. 201, Canada, December 2001.

7. Bar-Shalom, Y., X. R. Li and K. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, John Wiley, New York, 2001.

8. Kalman, R.E., *"A new approach to linear filtering and prediction problems", Transactions of the ASME – Journal of Basic Engineering*, No. 82 (Series D), pp. 35-45, 1960.

9. Power, P. W. and J. A. Schoonees, *"Understanding background mixture models for foreground segmentation", Proceedings of Image and Vision Computing,* New Zealand 2002.

10. Javed, O., M. Shah, *Automated Multi-Camera Surveillance Algorithms and Practice, Springer*, New York, 2008.

11. Withagen, P. G., *Object Detection and Segmentation for Visual Surveillance*, University of Amsterdam, Ph.D. Thesis, 2005.

12. Wang, L., W. Hu and T. Tan, *"Recent developments in human motion analysis",* Pattern Recognition, Vol. 36, No. 3, pp. 585-601, 2003.

13. Cutler, R. and L. Davis, *"View-based detection," Proceedings of Fourteenth International Conference on Pattern Recognition*, Vol. 1, pp. 495-500, Brisbane, Australia, August 1998.

14. Yilmaz, A., O. Javed and M. Shah, *"Object tracking: A survey",* ACM Computing Surveys, Vol. 8, Issue 4, 2006.

15. Jabri, S., Z. Duric, H. Wechsler and A. Rosenfeld, *"Detection and location of people using adaptive fusion of color and edge information", Proceedings of International Conference on Pattern Recognition*, 2000.

16. Liyuan, L. and L. Maylor, *"Integrating intensity and texture differences for robust change detection" IEEE Transactions on Image Processing*, Vol. 11, No. 2, pp. 105–112, February 2002.

17. Ridder, C., O. Munkelt and H. Kirchner, *"Adaptive background estimation and foreground detection using Kalman filtering", Proceedings of International Conference on Recent Advances Mechatronics*, pp. 193-199, 1995.

18. Wren, C. R., A. Azarbayejani, T. Darrell and A. P. Pentland, *"Pfinder: Real-time tracking of the human body", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 780–785, July 1997.

19. KaewTraKulPong, P. and R. Bowden, *"An improved adaptive background mixture model for realtime tracking with shadow detection", Proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01, Computer Vision and Distributed Processing,* September 2001.

20. Comaniciu, D., V. Ramesh and P. Meer, *"Real-time tracking of non-rigid objects using mean shift", International Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 142–149, South Carolina, 2000.

21. Shan, C., T. Tan and Y. Wei, *"Real-time hand tracking using a mean shift embedded particle filter",* Pattern Recognition, Vol. 40, Issue 7, pp. 1958-1970, 2007.

22. Comaniciu, D., V. Ramesh and P. Meer, *"Kernel-based object tracking", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 5, pp. 564–575, May 2003.

23. Comaniciu, D., V. Ramesh and P. Meer, *"Mean shift and optimal prediction for efficient object tracking", International Conference on Image Processing*, Vol. 3, pp. 70–73, Vancouver, BC, Canada, 2000.

24. Bradski, G. R., *"Computer vision face tracking for use in a perceptual user interface", Proceedings of IEEE Workshop on Applications of Computer Vision*, pp. 214–219, Princeton, NJ, 1998.

25. Tissainayagam, P. and D. Suter, *"Visual tracking with automatic motion model switching",* Pattern Recognition, Vol. 34, pp. 641–660, 2001.

26. Paragios, N. 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, 2000.

27. Isard, M. and A. Blake, *"Condensation - conditional density propagation for visual tracking",* International Journal of Computer Vision, Vol. 29, No.1, pp. 5–28, 1998.

28. Breidt, F. J. and A. L. Carriquiry, *"Highest density gates for target tracking", IEEE Transactions on Aerospace and Electronic Systems*, Volume: 36, Issue 1, pp. 47-55, January 2000.

29. Karlsson, R., *Simulation Based Methods for Target Tracking,* Licentiate Thesis, Department of Electrical Engineering, Linköping University, Linköping, Sweden, 2002.

30. Kirubarajan, T. and Y. Bar-shalom, *"Probabilistic data association techniques for target tracking in clutter", Proceedings of the IEEE*, Vol.92, Issue 3, pp. 536–557, 2004.

31. Cuzol, A. and E. Memin, *"A stochastic filtering technique for fluid flow velocity fields tracking", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, No. 7, pp. 1278-1293, July 2009.

32. Wiener, N., *The Extrapolation, Interpolation, and Smoothing of Stationary Time Series with Engineering Applications,* John Wiley, New York, 1949.

33. Brown, R. G. and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises and Solutions 3 ed., John* Wiley & Sons Inc., New York, 1997.

34. Simon, D., *"Kalman Filtering, Embedded Systems Programming*, pp. 72-79, June 2001.

35. Maybeck, P. S., *Stochastic Models, Estimation, and Control*, Academic Press, New York, 1979.

36. De Villiers, H. B., *Correlation and Tracking using Multiple Radar Sensors*, MSc. Thesis, Stellenbosch University, 2005.

37. Brookner, E., *Tracking and Kalman Filtering Made Easy*, John Wiley, New York, 1998.

38. Särkkä, S., *Recursive Bayesian Inference on Stochastic Differential Equations*, Ph.D. Thesis, Helsinki University of Technology, Finland, 2006.

39. Welch, G. and G. Bishop, *An Introduction to the Kalman Filter*, University of North Carolina at Chapel Hill, Chapel Hill, NC, 1995

40. Simon, D., *Optimal State Estimation,* John Wiley & Sons Inc., Hoboken, New Jersey, 2006.

41. Mazor, E., A. Averbuck, Y. Bar-Shalom and J. Dayan, *"Interacting multiple model methods in target tracking: A survey", IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, pp. 103–123, January 1998.

42. Blom, H. A. P., *"An efficient filter for abruptly changing systems", Proceedings of 23rd IEEE Conference on Decision and Control,* Las Vegas, December 1984.

43. Bloomer, L. and J.E. Gray, *"Are more models better?: The effect of the model transition matrix on the IMM filter", Proceedings of the Thirty-Fourth Southeastern Symposium on System Theory*, pp. 20-25, 2002.

44. Wang, X., S. Challa and R. Evans, *"Gating techniques for maneuvering target tracking in clutter", IEEE Transactions on Aerospace and Electronic Systems*, Vol. 38, No. 3, pp. 1087-1097, July 2002.

REFERENCES NOT CITED

Piccardi, M., "*Background subtraction techniques: a review*", *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4, pp. 3099-3104, October 2004.

Bailo, G., M. Bariani, P. Ijas and M. Raggio, "*Background estimation with Gaussian distribution for image segmentation, a fast approach*", *Proceedings of IEEE International Workshop on Measurement Systems for Homeland Security, Contraband Detection and Personal Safety Workshop*, pp. 2-5, March 2005.

KaewTrakulPong, P. and R. Bowden, "*A real time adaptive visual surveillance system for tracking low resolution color targets in dynamically changing scenes*", Image and Vision Computing, Vol. 21, pp. 913-929, 2003.

Friedland, B., "*A review of recursive filtering algorithms*", *Proceedings of AFIPS Joint Computer Conferences in Spring Joint Computer Conference*, pp. 163-180, Atlantic City, New Jersey, May 1972.

Wuest, H., "*Dynamic tracking and data association in image sequences*", Licentiate Thesis, Mannheim University, 2004.