OCCLUSION-AWARE 3D MULTIPLE OBJECT TRACKING FOR VISUAL SURVEILLANCE

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

BY

OSMAN TOPÇU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2013

Approval of the thesis:

OCCLUSION-AWARE 3D MULTIPLE OBJECT TRACKING FOR VISUAL SURVEILLANCE

submitted by **OSMAN TOPÇU** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

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

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** _____

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

Assist. Prof. Dr. Ali Özer Ercan
Co-supervisor, **Electrical and Electronics Engineering Dept., Özyeğin University** _____

Examining Committee Members:

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

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

Prof. Dr. Orhan Arıkan
Electrical and Electronics Engineering Dept., Bilkent Univ. _____

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

Assoc. Prof. Dr. Umut Orguner
Electrical and Electronics Engineering Dept., METU _____

Date: _____

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

Name, Last Name: OSMAN TOPÇU

Signature :

ABSTRACT

OCCLUSION-AWARE 3D MULTIPLE OBJECT TRACKING FOR VISUAL SURVEILLANCE

Topçu, Osman

Ph.D., Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. A. Aydın Alatan

Co-Supervisor : Assist. Prof. Dr. Ali Özer Ercan

September 2013, 81 pages

This thesis work presents an occlusion-aware particle filter framework for online tracking of multiple people with observations from multiple cameras with overlapping fields of view for surveillance applications. Surveillance problem involves inferring motives of people from their actions, deduced from their trajectories. Visual tracking is required to obtain these trajectories and it is a challenging problem due to motion model variations, size and illumination changes and especially occlusions between moving objects. By the expense of increasing number of cameras, tracking in 3D world coordinates is preferred over its 2D counterpart due to decreased viewpoint dependency and better occlusion handling through ordering of the targets. In this novel algorithm, our contributions are (1) observation error is increased in accordance with the estimated occlusion probability so that uncertainty in object location during any occlusion is taken into consideration, (2) increased observation error causes particles to expand, that in turn widens the association gate so that location uncertainty during occlusion is included into the association gate decreasing the chance to lose the object during occlusion, (3) observation error is allowed to change by the dimension of the associated silhouette so that distant and close objects can be tracked without parameter adjustment, while the particles are also saved from degeneration, when silhouettes are merged during occlusion. The improved performance of the proposed tracker is demonstrated in comparison with other state-of-the-art trackers using PETS 2009, as well as EPFL and PETS 2006 datasets. Furthermore, the proposed algorithm is extended for the surveillance systems whose cameras are not synchronized in time. Time difference between cameras are estimated by Gaussian mixture kernel density estimator and compensated by particle filter trackers. The experiments indicate that the proposed algorithm is able to work, when the amount of time difference between cameras is within one second. Finally, the position and velocity states of the proposed algorithm is

divided into linear and nonlinear parts in a Rao-Blackwellized fashion. In this formulation, velocity is marginalized by a Kalman filter, while the object position is filtered through a particle filter. It is shown that the resulting algorithm performs competitively, when number of particles are reduced without performance degradation.

Keywords: Surveillance, 3D Visual Tracking, Point Object Tracking, Occlusion, Particle Filter, Appearance, Multi-Camera, Multi-Target, Time Synchronization, Rao-Blackwellization, Marginalization

ÖZ

GÖRSEL GÖZETLEME AMAÇLI KAPANMA GÖZETEN 3 BOYUTLU ÇOKLU NESNE TAKİBİ

Topçu, Osman

Doktora, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. A. Aydın Alatan

Ortak Tez Yöneticisi : Yrd. Doç. Dr. Ali Özer Ercan

Eylül 2013 , 81 sayfa

Bu tez çalışması, örtüşen görüş alanına sahip çok sayıda kameradan gelen gözlemlerle çok sayıda insanı anında takip etmek için kapanma gözeten bir parçacık süzgeci çerçevesini sunmaktadır. Gözetleme problemi; insanların güdülerini, gezingelerinden ortaya çıkarılan eylemlerinden çıkarmayı içerir. Görsel takip, bu gezingeleri elde etmek için gereklidir ve hareket modeli çeşitliliği, nesne boyutu ve aydınlanma değişimleri ve özellikle hareketli nesneler arasındaki kapanmalardan ötürü zor bir problemdir. Giderek artan sayıda kameraya harcama yapılmasıyla 3 boyutlu dünya koordinatlarında takip yapmak, görüş açısına daha az bağlı kalınması ve hedefleri uzaklıklarına göre sıraya dizerek daha iyi kapanma yönetimi yapılması sebepleriyle 2 boyutlu benzerlerine tercih edilmektedir. Bu yeni algoritmada yapılan katkılar (1) gözlem hataları kapanma olasılıklarına göre arttırılmaktadır; böylece kapanma sırasında nesne pozisyonundaki belirsizlik hesaba katılmaktadır, (2) arttırılan gözlem hataları parçacıkların yayılmasına sebep olur; bu da eşleme aralığının genişlemesine sebep olur ki böylece kapanma sırasında nesneyi kaybetme riski azalır, (3) gözlem hatasının eşlenen silüetin boyutu ile değişmesine izin verilmektedir böylece hem uzaktaki ve yakındaki nesneler değişken ayarı yapmadan takip edilirler hem de parçacıklar kapanma sırasında silüet birleşmelerinden dolayı dejenere olmazlar. Önerilen takipçinin iyileştirilmiş başarımı, tekniğin son durumu olan takipçilerle PETS 2009, EPFL ve PETS 2006 veri kümeleri kullanılarak karşılaştırmalı olarak gösterilmektedir. Ayrıca, önerilen algoritma, kameraları eşzamanlı olmayan gözetleme sistemleriyle çalışabilecek şekilde genişletilmektedir. Kameralar arasındaki zaman farkı Gauss karışımı çekirdek yoğunluk kestiricisi ile kestirilmekte ve parçacık süzgeçleri tarafından telafi edilmektedir. Deneylere göre, önerilen algoritma kameralar arasındaki zaman farkı 1 saniyeden az olduğunda çalışabilmektedir. Son olarak, önerilen algoritmanın konum ve hız durumları, Rao-Blackwell tekniğinde olduğu gibi, doğrusal olan ve

olmayan olarak ikiye ayrılmıştır. Bu formülasyonda, hız Kalman süzgeci ile marjinalleştirilirken, konum parçacık süzgeci ile süzülmüştür. Ortaya çıkan algoritmanın parçacık sayısı azaltıldığında, başarımı azalmadan rekabet edebilir başarım sergilediği gösterilmektedir.

Anahtar Kelimeler: Gözetleme, Görsel Takip, Örtüşme, Kapanma, Parçacık Süzgeci, Görünüş, Çoklu Kamera, Eşzamanlama, Rao-Blackwell

To my family

ACKNOWLEDGMENTS

Special thanks to Prof. Dr. Aydın Alatan for his valuable guidance and Assist. Prof. Dr. Ali Özer Ercan for his excellent support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CCTV	Closed-circuit television
CRA	Cumulative running average
GMMKDE	Gaussian mixture kernel density estimator
PF	Particle filter
EKF	Extended Kalman filter
UKF	Unscented Kalman filter
RMS	Root-mean-square
NTP	Network time protocol
GPS	Global positioning system

CHAPTER 1

INTRODUCTION

1.1 Introduction

Surveillance cameras are largely used by governments, law enforcement and shop owners. They are so widely used that no matter to which direction we turn our head in a busy street, we see a camera monitoring us. Video data from these cameras are collected, stored and displayed in a monitoring room. Human observers watch these cameras to detect any hazardous or illegal activities. In this respect, video surveillance can be defined as the monitoring of public places, transportation systems, factories and workplaces to detect illegal activities and to ensure public safety. As more and more areas are monitored, the required number of cameras increases, whereas the increasing number of cameras makes it difficult to effectively monitor those areas.

1.2 Evolution of Surveillance Systems

The first generation surveillance systems involve analog closed-circuit television (CCTV) systems [1]. These systems are composed of a couple of cameras that are connected to the monitors in a control room. A person has to sit in the control room and watch the monitors in such systems. These systems store analog data using analog storage techniques [1]. The stored data can be used to solve an incident by security forces after it happens. The main problem with the first generation surveillance systems is the presence of many cameras while limited number of eyes are monitoring them [2].

More than one million CCTV cameras were present in United Kingdom in 2003 [2]. Excessive number of cameras causes video information overload and thus makes it difficult to manage those cameras [3]. Moreover, making human observers to monitor these cameras is costly whereas its effectiveness is questionable. Research shows that human attention drops to unacceptable levels, after looking at a screen for about 20 minutes [3]. For these reasons, automation of surveillance systems becomes a must.

By the availability of digital storage and compression techniques, the semi-automatic second generation surveillance systems came to surface [1]. These systems can do computer vision tasks such as change detection, tracking and event analysis to aid their operators. The monitoring efficiency increases with the second generation systems. However, the necessity for robust algorithms stands still.

Surveillance cameras can monitor a limited area within their field of views. For the surveillance of larger areas, more cameras are required. Third generation surveillance systems are composed of a network of cameras installed to monitor a large area [1]. Third generation systems extend the capabilities of second generation systems by processing information from multiple cameras. The network topology of these visual sensors can be centralized, distributed or decentralized. A central processing unit gathers and processes all the information from all cameras within the centralized network. In distributed camera networks, each node processes its own data and whispers its findings to nearby nodes. Decentralized camera networks are composed of centralized networks of several cameras whose cluster heads are networks in a distributed fashion. It is foreseen that future surveillance systems are going to be composed of decentralized wide area camera networks as this topology is scalable [1].

The research on surveillance systems and algorithms are ongoing. Future systems should work properly under quite little or no supervision. Moreover, their performances should not depend on viewing angle of cameras. These systems should get more intelligent so that they can predict activities before they happen.

1.3 Motivation

The main motivation is to design surveillance systems that can detect *bad* motives. This detection can be performed by tracking objects. Performance of detection of these motives depends largely on the tracking performance. Therefore, these systems require reliable tracking. Tracking performance is largely affected by occlusions, especially dynamic occlusions of objects. Therefore, motivation of this thesis is to track objects by taking occlusions into consideration.

Consider a target traveling from location A to location B. There is a certain motive behind this action. If A is home and B is work, then the motive is to go to work. Actions are observable but motives are not observable; they are hidden. Given a particular action, "what is the probability that the target has a certain motive?" is a more reasonable question than "what is the motive of the target?". Motives and corresponding actions can be modeled as a random process. Motives are the random variables and actions are the realizations of the random process. Actions can be determined from trajectories of targets. It could be argued that 3D trajectories contain more information for better estimation of the actions compared to 2D trajectories. In addition, 2D trajectories depend on the camera viewpoint. Those surveillance systems that are trained by using 2D trajectories require re-training each time a new camera is installed with a slightly different viewpoint. This increases installation time and effort. These trajectories are generated by tracking targets. For this reason, 3D tracking has certain advantages over 2D tracking. To sum up, targets are tracked to generate trajectories and trajectories are used to determine actions that helps estimate the motives behind these actions.

Target tracking is a difficult problem due to occlusion, illumination changes and shadows. Occlusion limits performances of tracking algorithms as target correspondences cannot be obtained reliably. In case of occlusion, tracker might lose the target or mistake it for another. Therefore, trackers should be aware of occlusions so as to reduce tracking failures. Occlusion-awareness can be defined as taking actions during occlusions such that target is not mistaken for another or lost and increased observation error is taken into account. Hence occlusion-awareness is expected to improve the overall tracking performance.

For tracking targets in 3D world coordinates, more than one camera viewing the same scene is required. These cameras should observe the same target at the same time in order to locate its 3D coordinates. Cameras that are synchronized in time produce observations at the same time instant. As targets are tracked in 3D coordinates to yield 3D trajectories, time synchronization of cameras also become important.

1.4 Scope of the Study

The scope of this thesis is to track multiple targets in multiple cameras. The number of cameras cannot be excessive in typical surveillance scenarios; therefore, number of cameras are constrained to be equal to 2 that is the minimum number required for tracking objects in 3D coordinate frame. Two major problems associated with tracking is addressed in this study. Occlusion of targets during their tracking is the first problem, whereas the second one is tracking targets with unsynchronized cameras. In addition, Rao-Blackwellization strategy to improve sampling efficiency of the particle filter algorithm is addressed.

1.5 Contributions

The first contribution is a novel occlusion-aware particle filter tracking algorithm where occlusion information is incorporated into the particle filtering framework. The observation noise of particle filters are set proportional to the silhouette size so that small and large as well as distant and close objects can be tracked without adjusting the parameters. This formulation also provides performance improvement during occlusions, since silhouette size increases during occlusions as a result of merging of silhouettes. The performance increase is due to providing sufficient amount of observation uncertainty to trackers. In addition, observation noise is made proportional to the occlusion probability that provided additional observation uncertainty in such complicated occlusion cases. The second contribution is a novel algorithm for tracking using unsynchronized camera networks. Time difference between cameras are estimated and compensated within the proposed occlusion-aware tracking algorithm. Finally, Rao-Blackwell like formulation of the occlusion-aware particle filter tracking algorithm is also accomplished.

1.6 Outline of the Thesis

Chapter 2 is mainly devoted to the state-of-the-art in video object tracking and related concepts. An introduction to tracking is achieved in Section 2.1. A taxonomy of tracking algorithms is presented in Section 2.2, whereas Section 2.4 introduces the occlusion problem. Comparison of 2D tracking with 3D tracking and dynamic model based tracking is covered in Sections 2.5 and 2.6. Related work on video tracking is mentioned in Sections 2.3 and 2.7 and related work on occlusion handling is presented in Section 2.8. Chapter 3 explains the proposed occlusion-aware particle filter tracking algorithm. Chapter 4 details the algorithm developed for tracking targets with unsynchronized camera networks. Rao-Blackwellization of the proposed algorithm is explained in Chapter 5. The work performed in this thesis is concluded in Chapter 6. Sequential Bayesian filtering, the Bayes filter and Monte Carlo sampling

approximation is explained in Appendices A, B and C.1, respectively. Importance sampling, derivation of particle filter algorithm and variance reduction is covered in Appendices C.2, C.3 and C.4, respectively.

CHAPTER 2

LITERATURE REVIEW

2.1 What is Tracking?

Tracking is the act of locating a non-stationary object along the course of its action. In ball games, players track the ball; especially, goalkeepers do to save a goal. A simple example to understand tracking and related concepts is the problem of sailing at night without any navigation equipment (Figure 2.1). Consider we have to sail at night in open sea such that we have a rough idea of where we are. We only know that we have to sail north. In addition, there is no reference location to determine our position. Waves and sea streams are dragging the boat and wind is supplying the necessary force to sail. The only source of observation is the north star that is visible, if the sky is open. In order to sail north, we look at the north star, figure out which direction north is and steer accordingly. Meanwhile, sea waves and streams are deflecting the boat from its course. After a while, we look at the north star again and see that we have been deflected and correct our direction accordingly. We will repeat these procedures until we get to the destination. In this scenario, the north star is the observation, wind power makes our boat gain speed and sea streams and waves are the noises. There might be times when north star cannot be observed due to occluding clouds. At these times, we have to trust our sense of direction developed so far. This problem is similar to the tracking problem, except that we are trying to determine where the target is in the tracking problem rather than where we are.

Figure 2.1: A simple example to clarify tracking operation

2.2 A Taxonomy of Single-View Tracking Algorithms

Video tracking is the process of locating moving objects over video sequences [4]. Video tracking of multiple moving objects is a difficult problem due to dynamic occlusions of objects. Approaches in the literature can be broadly classified as vision-based techniques [5, 6, 7] and dynamic system model-based techniques [8]. Vision-based techniques mainly approach video tracking as finding correspondences across sequences, whereas dynamic system model based techniques consider it as an inference of target state given a set of incomplete observations. Vision-based techniques can also be sub-grouped as appearance/template [7], interest point [10, 11], and kernel-based methods [5, 12]. The appearance can be color histogram, dominant color, intensity template, textural template and eigentemplate. The interest points can be corners [9], SIFT [10] and SURF [11] features. Mean-shift tracking [12] is an example of kernel-based tracking methods. Dynamic system model-based techniques, on the other hand, can be sub-grouped as position-based, appearance-based and model-based approaches. Position-based methods can further be divided as 2D-position and 3D-position based techniques. There are also occlusion-aware approaches within the appearance-based methods [8].

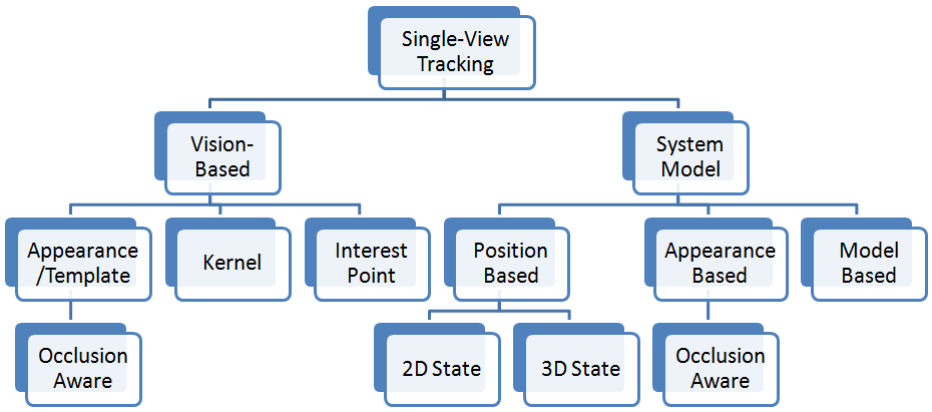


Figure 2.2: Single-view tracking literature overview

2.3 A Taxonomy of Multi-View Tracking Algorithms

Multi-camera tracking strategies differ in their approaches against detection and tracking subproblems (Fig. 2.3). One main class is the fusion approach, which focuses more on detection than tracking; therefore, it is more suited to high density scenarios. In fusion-based techniques [13, 14, 15, 16], the detections in each view is mapped to a common view, such as a common scene plane, to obtain better detection performance. In this set of techniques, the number of cameras and their placement [15] becomes important for making accurate detections and reducing false detections. The required common view is usually selected as the ground plane, since objects are assumed to be residing on a ground plane. However, this assumption that reduces the usability of these approaches. Fusion approaches can be further divided as measurement-level fusion and tracker-level fusion [17]. Measurement-level approaches fuse observations first and then applies tracking, whereas tracker-level fusion approaches first track

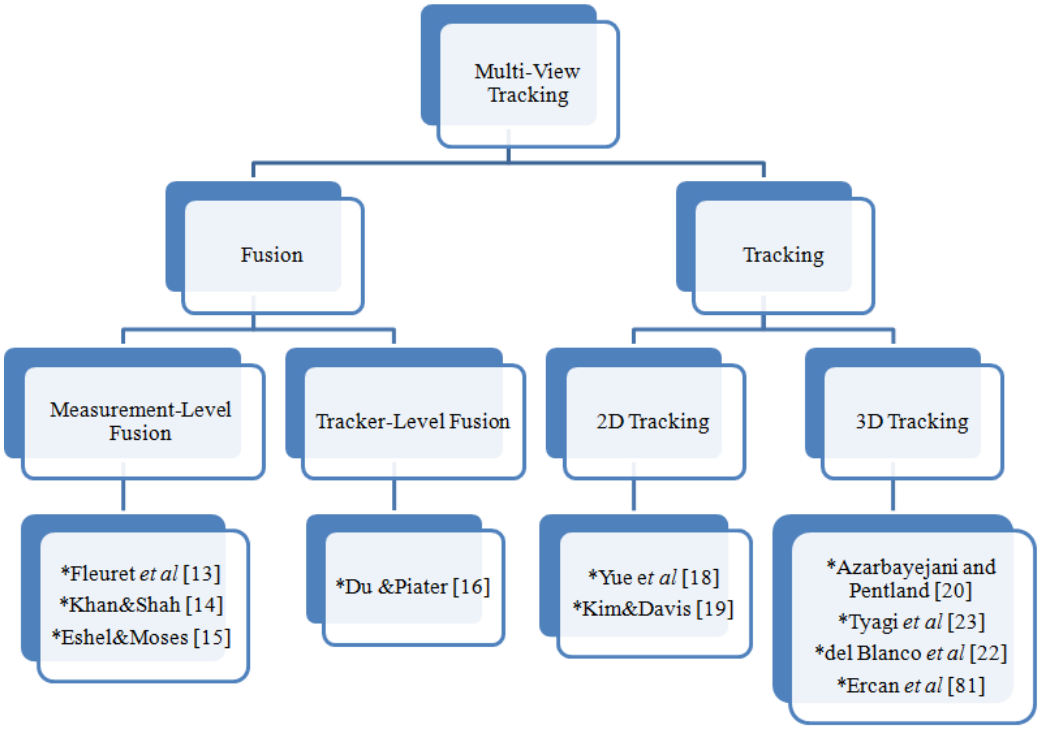


Figure 2.3: Main approaches to multi-view person tracking problem in the literature.

targets in different views, and then fuses these tracking results. Occupancy is the main feature used in measurement-level techniques and they are mainly batch algorithms processing a number of consequent frames at a time. As a measurement-level method, Khan and Shah [14] maps foreground likelihoods of pixels to a reference view, where all likelihoods are accumulated to detect targets. However, frame-rate and target velocity must be arranged such that foreground regions overlap in time. Moreover, there is an enormous amount of communication between cameras, since whole foreground likelihood image is transmitted to the reference view at every time step. It should be noted that a group of frames are processed at a time to construct trajectory worms, the resulting resolution should be low enough to catch up with a suitable frame rate. As another measurement-level method, Fleuret et al [13] divides ground plane into grids, so that only one person can fit in a grid. Foreground images are generated from a batch of frames. Joint probability of observations and trajectories are computed by using color model, ground plane occupancy and motion model; next, Viterbi algorithm is applied to obtain the trajectories. All the measurement-level fusion methods require perfect synchronization of frames and might yield false positives with only two views.

As a prominent example to tracker-level fusion techniques, Du and Piater [16] use individual particle filters in each camera and ground plane to track targets. The ground plane particle filter fuses information from the other particle filters in the views. The filtered state, which is the principal axis of people silhouettes, is sent back to the filters in each camera where a final state is generated for each view. The requirement of $M + 1$ particle filters per object in a multi-camera setting of M cameras is not computationally feasible. As opposed to the proposed method, there is no occlusion handling strategy and their results do not contain scenes with frequent occlusions.

The other main class for multi-view techniques is the tracking branch, as in Figure 2.3. This class of approaches usually use filtering techniques for tracking in which observations from multiple views are associated to trackers. This class can further be divided into 2D tracking and 3D tracking approaches. 2D tracking approaches are mainly extension of 2D image tracking to cope with occlusions and tracking is performed on common view that can be the ground plane. For example, Yue [18] is employing single view trackers to track targets in both views; ground plane homography between views is used, when target is occluded in one view. Such techniques usually assume that there is no simultaneous occlusion in both cameras. Kim and Davis [19] propose another example of 2D approach in which they extract central vertical axis of human blobs and map these lines to ground plane by using homography where these lines are intersected to compute ground positions of people. Next, the ground positions are tracked by using particle filters for each person. As person silhouettes might merge or people occlude one another, central axes might become corrupted that yield performance decrease during tracking [19].

3D visual tracking algorithms dates back to Azarbayejani's work [20], in which blob features are utilized to self-calibrate stereo cameras that are translated and rotated in z-axis only. Using the calibration information, the blobs are tracked in 3D coordinates by an extended Kalman filter; however, occlusions are completely ignored in their approach. On the other hand, Black and Ellis [21] applied background subtraction to 2 or 3 views to obtain blob centroids and blob centroids are triangulated to come up with the estimation of 3D target position that is tracked by 3D Kalman filter. The performance of their algorithm [21] in denser environments, where targets maneuver and occlude one another, is not stated. Moreover, their algorithm is vulnerable to association errors and long-term occlusions; Kalman filter also perform poorly, if objects that are moving forward turn back and move the opposite direction. The method due to del Blanco [22] proposes a 3D particle filter tracking algorithm using silhouette and color information. Each particle has an appearance state in addition to position and motion state and the particles are considered to occupy a differential spherical volume. In their approach, the particles are spread within a volume supposed to belong to a target and particle weights are raised according to silhouette, if their projections onto image planes correspond to foreground pixels. However, no explicit occlusion handling is proposed and color appearance has to be almost constant during the course of target motion. As another 3D visual tacking algorithm, Tyagi et al [23] forms the target state in 3D coordinates and target appearance that is the color distribution in all views. The next target state is computed by using 3D mean-shift iteration over all views, whereas the observations are related (fused) as they match with target appearance. However, their method strictly requires at least 2 unoccluded view to be available over 3 views at a particular time [23].

2.4 Occlusion Problem in Visual Tracking

Occlusion, as a computer vision term, is the situation where an opaque object prevents another to be observed from a camera's point of view and it is a major problem of visual tracking. In order to be free of occlusion, the cameras should be placed upside down that results in monitoring a much smaller area when compared to other configurations. In addition, placement of cameras as desired is not possible all the time due to scene constraints. For these reasons, occlusion stands as a major problem in visual tracking. Typically all the vision-based tracking methods fail to find any correspondences when the target is occluded; or worse, they might come up with a false correspondence due to a similar looking object. Dynamic system model-

based methods use the learned motions of targets to infer their locations when targets undergo occlusion. If target is occluded for a short time, these methods are able to recover it shortly as soon as the target reappears, which is called "tunneling" [24]. Therefore, dynamic system model based methods are advantageous over vision based methods for their "tunneling" ability. Moreover, Mittal and Davis [25], Khan and Shah [26] and Kim and Davis [19] argue that multiple cameras monitoring the same scene should be used to cope with a high density of people in the scene who are dynamically occluding one another. Hence, if an object is occluded in one view, observation from another un-occluded view can be used for tracking.

2.5 2D Tracking versus 3D Tracking

2D tracking is finding locations of objects in image coordinates whereas 3D tracking is locating them in 3D world coordinates. 3D tracking requires at least 2 cameras and their calibration with respect to a common world coordinate system. On the contrary, 2D tracking requires neither multiple cameras nor their calibration. However, 2D tracking depends on the viewpoint of the camera; if the target is occluded, then its tracker cannot get proper observation. When the target travels away from a camera, its image on the image plane shrinks. It enlarges when that object travels towards the camera. This scale change creates performance limitations for 2D kernel based trackers [23]. However, 3D tracking does not experience performance limitation due to scale change and does not depend on the camera viewpoint as much as 2D tracking [23]. In addition, there could be an un-occluded view at any time in 3D tracking with multiple cameras.

2.6 Dynamic System Model Based Tracking

Dynamic model based 3D tracking methods emerge as a reasonable choice for tracking multiple objects experiencing dynamic occlusions. Dynamic model based tracking methods consider the sequence of target states as a random process. Likewise, these algorithms model the sequence of target observations as a random process. The widely acknowledged dynamic model based tracking method is Kalman filter, which is an optimal estimator when state and observation processes are linear and Gaussian [27]. In visual tracking, observations are generated by the projection of 3D objects onto the 2D image plane and this projection is a nonlinear operation. Therefore, observation process is nonlinear which makes Kalman filter non-optimal. Besides, targets might make sharp turns or instantly speed up that makes state process nonlinear as well. Extended Kalman filter is an *ad hoc* filter [27] that linearizes the nonlinear state and observation equations around the current state estimate using Taylor expansion. The extended Kalman filter's performance depends on the degree of the nonlinearity [28]. In addition, it cannot handle those cases when observation likelihood is multi-modal [28], that is, the presence of similar looking objects nearby [29]. On the other hand, particle filters do not exhibit these limitations and can handle nonlinear and non-Gaussian state and observation processes [30] as well as multi-modal observation likelihoods. Particle filters sample the state space with Monte Carlo samples. Moreover, it is relatively easy to incorporate new information to the filtering equations. For these reasons, particle filters are used for tracking targets in this thesis, as we incorporate occlusion information into the particle filtering framework.

2.7 Related Work on Video Tracking

An overview of tracking and related concepts has been written so far. Object tracking literature can be categorized according to target representation, target detection and tracking methods. Visual tracking literature is schematically illustrated in Figure 2.2. As far as surveillance is concerned, the target of interest is primarily people and/or vehicles. There are computer vision approaches that consider targets as a set of interest points [31, 32, 9]. They track corners on the objects using optical flow [32] and Kalman filter [31]. Since each object region might contain a set of interest points, they are grouped according to their proximity and motion. Some work in the literature represent targets using parameterized contour lines. Leeds people tracker [33] tracks pedestrian silhouette contours using Kalman filter. In the paper by Isard and Blake [24], parameterized contour lines are tracked using conditional density propagation algorithm which is similar to particle filter. Wang [34] uses spatial layout and color distributions of each color mode within the image region corresponding to a target. Jepson [35] uses a wavelet-based appearance model. Bradski [5], Comaniciu [12] and Han [36] propose variants of kernel based target representation and tracking algorithms. Han [36] proposes an appearance model that is the probability density function of color features composed of Gaussian mixtures. Comaniciu [12] tracks image regions with mean-shift algorithm, while Bradski [5] tracks by a similar cam-shift algorithm. McKenna [37] represents and tracks targets using adaptive Gaussian mixtures of their color in Hue-Saturation color space. Perez [38] represents targets using their Hue-Saturation-Value (HSV) color histograms and tracks them with particle filter. Senior [39] represents targets using a template of grayscale intensity and a probability mask of the grey levels. Gabriel [40] defines the objects as "blobs" that are image regions and keeps track of their creation, deletion, merge and split.

2.7.1 Video Object Detection Methods

Automatic video object detection methods mainly are based either background subtraction or optical flow. Optical flow methods are sensitive to image noise and thus less preferred. Meyer [41] uses optical flow to initialize a contour based tracking algorithm. Background subtraction is a widely acknowledged detection technique. Various background subtraction algorithms exist in the literature. Stauffer [42] comes up with a background model based on spectral features. Stauffer models a pixel's intensity values with a mixture of Gaussian probability density function. A pixel's intensity is classified as background depending on its evidence and persistence. Lee [43] modifies this algorithm, so that its convergence speed and model accuracy are increased. This modified algorithm yields false alarms due to auto-iris of the camera. The frame differencing concept is incorporated into the modified algorithm so that false alarms due to auto-iris of the camera are reduced. This background subtraction algorithm is used in this thesis. Chao [44] computes background image using mode of the pixel intensities. Zhang [45] uses median of R, G and B channels of each pixel to construct a background image. Lai [46] keeps two background images; mode image and running average image. If intensity change for a pixel is large, running mode algorithm is used for background subtraction; otherwise, running average algorithm is used. They are simple yet require memory. Lim [47] expands the work of Ivanov [48] to come up with a background subtraction algorithm using two cameras with overlapping field of views. They use depth information of the scene for background subtraction. The disparity between two cameras corresponding to a foreground pixel is different than a background pixel. Exploiting this idea, Lim [47]

reported that fast illumination changes like headlights of a starting car at night are handled. Krumm [49] uses depth and color information for object detection using a stereo camera pair. Javed [50] performs pixel, region and frame level processing for more accurate background subtraction method. Pixel level processing involves color based subtraction that is mixture of Gaussian based algorithm of Stauffer [42] and gradient based subtraction. Their gradient based subtraction involves modeling gradient between the pixel and its neighbor according to Rayleigh distribution. Region level processing is to search for high gradient to determine object boundaries. Frame level processing comes into play, if more than 50% of the frame is labeled as foreground by color based background subtraction method. Only gradient based method is used in this case. Region level processing of Javed [50] can be used for separating a merged foreground blob of multiple people.

2.8 Related Work on Occlusion Handling

There is a vast literature work on occlusion handling. Those among the 2D tracking approaches face with the problem of determining "who is occluding who?". 3D tracking approaches, on the other hand, can determine the occlusion inter-relationships by depth ordering. Those closer to the camera are less likely to be occluded whereas those far away are more likely. Apart from these, there are approaches that assume at least one non-occluded view in a multi-camera environment [51]. 2D tracking approaches [8, 52, 39] deduce occlusion relationships from appearances. Kwak [53] divides target region into cells and uses a classifier to determine whether each cell is occluded or not.

2.9 Discussion on Camera Placement

Ideally, two cameras are sufficient to extract 3D information of scene objects. Additional cameras bring extra information about the scene activities. The amount of information brought by additional cameras depends on their placement. As cameras are getting closer to each other, mutual information between them increases that means the extra camera provides no additional information. Conversely, as this angle increases, the extra camera brings more information. Figure 2.4 illustrates localization uncertainty of cross-section of a human shaped object in different camera configurations. In this figure, localization uncertainty is illustrated to decrease as the angle between cameras gets closer to 90°.

Similarly, laterally placed cameras experience the adverse effects of occlusion such as narrowed visibility. As the camera is raised and tilted towards the ground, these adverse effects diminish; thus, information content increases due to the raised visibility. The increased detail level of object observations promotes identification of objects in consequent frames because more discriminative information is inherent in object appearances.

Ercan [54] studied optimal camera placement problem. For the case when observation noise variance of cameras are the same, Ercan proved that uniform placement of cameras is optimal. For two cameras, this corresponds to placing cameras 90° apart. More discussions on the placement problem, when static occluders are present and cameras are not identical, are presented in [54].

2.10 Discussion on Benefits of Contextual Information

Visual tracking is a relatively difficult problem; occlusions, shiny or shadowy scene regions, complicated object behaviors, static occluders and many more create difficulties. Any additional information is strongly appealed to improve tracking performance. So far, the researchers have focused on algorithmic improvements. Procedures to strengthen weak sides of algorithms are the outcome of these studies. For example, the idea of running parallel Kalman filters each having a different process noise model allowed tracking of maneuvering objects [55]. Interacting multiple modes (IMM) is a multiple model technique in which state estimates and covariance matrices from multiple models are combined according to a Markov model [55]. However, quite few researchers have noticed the benefit from additional information from the scene elements. Structural elements of the scene such as roads, lanes, bridges, doors, gates and corridors have a great influence on the way humans behave. People drive vehicles on the road and follow lanes. By introducing information of scene structures to tracking algorithms, tracking performance is expected to rise. These information can be extracted by unsupervised algorithms, such as those proposed by Saleemi [56] and Anjum [57]. Saleemi [56] proposed an algorithm to learn patterns of optical flow using statistical techniques and Anjum [57] proposed a trajectory clustering algorithm. By the help of trajectory clusters and motion patterns, different process noise models can be generated and IMM filtering technique can be applied.

Alternatively, Maggio [58] proposes adding contextual information into a tracking algorithm

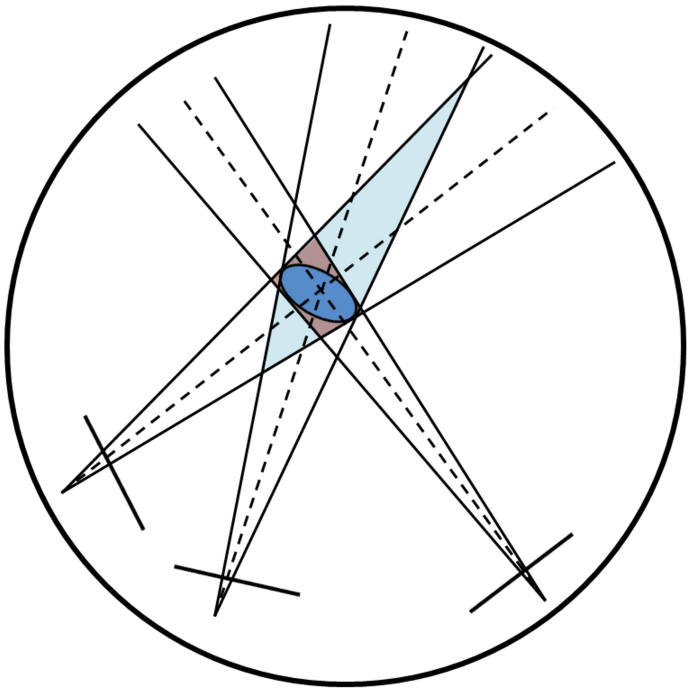


Figure 2.4: Different camera placements and their effects on localization. Cross-section of human-shaped object is shown in blue. The dashed lines indicate the lines emanating from silhouette centroids. The solid lines are drawn to illustrate the extents of silhouette errors. The resulting localization accuracy is best when cameras are placed at 90° with each other.

to improve its performance. Maggio [58] extracts target birth density and clutter density from the image sequence. Target birth density is generated from those image regions where targets are initialized and clutter density is formed using the detections that are not associated to any objects. By using this strategy, more focus can be exerted to detect objects in those regions where they likely to appear and more evidence can be sought in those cluttered image regions in order to become resistant to false targets.

To sum up, incorporating useful information to tracking algorithms improves their performances. Structural elements of the scene shape object behaviors and their integration into tracking algorithms raise their performances. In addition, parameters of tracking algorithms can be adjusted according to their states by learning where they are likely to fail and where they are likely to make detections.

2.11 Discussion on Literature Findings

To sum up, targets need to be tracked to generate their trajectories and these trajectories are used for estimating their actions. The motives of targets are predicted from these actions. Dynamic system model based tracking can perform better than computer vision based methods when target correspondences cannot be obtained. Occlusion is one of these cases where finding correspondences is difficult. Occlusion handling improves tracking performance as trackers are not affected from possible erroneous observations. Moreover, tracking targets in 3D coordinates has advantages over 2D tracking of them. 3D tracking is not as dependent on camera viewpoint as 2D tracking. Occlusion handling is much easier in 3D tracking when compared to 2D tracking. Furthermore, particle filters that are able to track targets with nonlinear and non-Gaussian state and observation processes, appear to generate better tracking performance than the Kalman filter and its modifications. In this thesis work, based on these findings, targets are tracked in their 3D coordinates by using particle filters such that occlusions are handled during tracking.

CHAPTER 3

APPEARANCE BASED OCCLUSION-AWARE TRACKING OF MULTIPLE TARGETS

3.1 Introduction

When a camera is viewing 3D objects within a scene, visible light reflects from the objects and the whole scene is projected onto the 2D image plane [29]; hence, only the 2D projections of objects onto the image plane can be observed. Although today direct 3D observation is possible via structured-light 3D scanning devices, their operating range is very limited and they cannot operate in direct sunlight, such as outdoors [60, 59]. However, 3D information about objects can be extracted by using observations from at least 2 or more cameras with overlapping fields of views. Moreover, 3D information allows the real situation in a scene to be better understood. For these reasons, 3D information is utilized in many application areas, such as surveillance, human-computer interaction, augmented reality, video indexing, sports analysis, 3DTV, etc. [61, 62]. It can be argued that by the extraction of 3D information from a scene, less ambiguous and more realistic information can be attained in any application, including visual surveillance and tracking.

In visual surveillance applications, multi-camera tracking is a medium of 3D information extraction and capable of locating objects using information from multiple sensors. A portion of these tracking methods considered controlled environments, such as smart rooms [20, 49, 63, 25, 66, 13, 22, 15] and sports facilities [14, 64] in which number of cameras and their placement can be arranged [65, 15]. Additionally, prior knowledge on the environment is available and assumptions, such as objects are residing on a common ground plane, can be stated. This prior knowledge can be exploited to quantize the scene into 2D grids [13], voxels [66] or stack of planes parallel to ground plane [14]. Indeed, tracking performance is a function of number of cameras, their placement, resolution, frame-rate [14], level of crowdness of the environment, occlusions, calibration and synchronization accuracy. In such environments, control over the number of cameras, their placement and resolution is possible. For example, Eshel and Moses [15] proposed an algorithm for tracking a dense group of people by using 3 to 9 surrounding cameras that are placed at 6m above the ground and tilted downwards more than angle of 45°; however, in practice, mounting cameras as described is not feasible. Since most of the multi-camera tracking methods are applicable to uncontrolled environments, such as parking lots, train stations [67] and school campuses [68, 21, 18], the number of cameras, their placement and resolution cannot be arranged as desired in such environments. Additionally, scene effects, such as shadows, illumination changes and background clutter creates extra difficulties. This limits the number of applicable algorithms in the literature and the

level of crowdness those applicable algorithms can handle. In summary, 3D information of a scene helps managing many problems in such uncontrolled environments.

As multiple cameras are required for 3D information extraction, relative positioning of cameras has also influence on the total amount of observed information. Information content increases as cameras are placed apart horizontally, as well as they are elevated and tilted downwards. The reason of this increase is the extended total field of view and the decrease in occlusion resulting from scene objects. For single view applications, the occlusions are unavoidable for any camera placement; however, multiple camera applications are expected to benefit from a possible unoccluded view.

Object representation is an important element in visual tracking. Considering 3D objects in the scene are projected to an image region, features of this region can be used to represent an object. As a feature, boundary or shape of a region is typically used for the representation [33, 24]. However, boundary of the object might still cause identification errors, if it is occluded or if objects with similar boundary are present nearby, which is usually the case, especially if the object is human. As another feature, intensity or gradient templates [53] of this region, as well as textures, such as local binary patterns [69], can also be used. In this case, the object needs to be rigid and a change in its projected pose should not make a considerable difference in the template. Even if the object is rigid, pose change might change the template values, for example, template size and values change temporally, when a vehicle is making a turn. As a different representation, interest points within this image region is another option for object description [31, 9, 10, 11]. However, sufficient number of robust interest points should be available, so that tracking can continue, despite self-occlusion of objects. Additionally, color histogram feature of the image region can be used to identify objects [38, 70, 23]. Color histogram is suitable for tracking nonrigid objects, such as people [38, 70, 36] and provides robustness against self-occlusions and temporal changes in the pose and scale. Linking image regions corresponding to objects in consequent frames can be accomplished by mean-shift algorithm [12, 36]. However, people with similar appearance or background regions with similar color content might create ambiguities. Moreover, intensity might vary across the scene, i.e., some portion of the scene might be shadowed in which case tracking fails. Yet another representation is a *blob* which is defined as an image region where one or more visual features, such as color, motion, texture, brightness is shared by all the pixels within the region and not shared by surrounding pixels [20]. Motion blobs (i.e. silhouettes) that can be created automatically by the help of background subtraction algorithms [42, 43] is quite a well-known object representation within the visual tracking community [42, 21]. Finally, projections of 3D object models can also be searched within an image region according to a cost function [65, 71, 72]. Object pose becomes important in this case and many projections of the model need to be evaluated depending on its pose. Moreover, more 3D models need to be created and evaluated for a variety of object types in heterogenous environments; more discussion on object representation is available in [73]. However, among all the representation techniques, silhouette (motion blob) is more appropriate for uncontrolled scenarios with static cameras and tracking motion blob centroid (point object model) results in a better 3D localization accuracy. Even calibration [20] and synchronization [74] of cameras can be accomplished by using such silhouettes of the scene objects.

After representation of moving objects, association of these representations over multiple simultaneous views is another problem for multi-view tracking [75]. The geometric relation between multiple views and the scene points [76] provide strong cues that can be used as a reasonable solution to this problem. Homography of the common scene plane in multiple

views or epipolar geometry are examples of such a relation. For the association problem, there are many correspondence finding algorithms in the literature. Some researchers take occupancy approach and find correspondences across multiple views through the 3D space the generative object occupies [14, 13, 15]. However, this approach might yield false associations, only when two cameras are available [14]. Some other researchers match multi-view observations according to the ground plane positions of objects [19, 16]; they extract central axis line of people and project them to ground plane where lines from the same person intersect at feet position. Reliable detection of objects become crucial, as occlusions create ambiguities in this approach. Appearance similarity between views is also used to associate observations [49, 23]. Unfortunately, this approach might be influenced by similar-appearance objects nearby. As a completely different association-free approach, all observation combinations are triangulated and tracked [63]; 3D positions that cannot be tracked reliably over long sequences are eliminated. The computational complexity of such an approach could be devastating in case of crowded scene with many cameras. In summary, it could be argued that for fixed cameras observing an arbitrary scene, epipolar geometry defined by the camera pairs should be exploited to obtain better association between views.

3.2 Proposed Multi-View Multiple Object Tracking Approach

In general, tracking can be defined as the process of generating state sequences of dynamic objects over time by using observations from sensors. Objects of interest have some form of internal state, i.e., motive, that cannot be observed directly. Their actions, i.e., motions, are motivated by this internal state. These motions are observable. Hence, the problem can be modeled as a random process where motives are the random variables and actions are the functions of time and the motive [77]. The sequence of object observations may contain noisy, missing and false observations. Instead of completely relying on observations as in [14, 13, 15], efforts are directed to estimate object states. In this respect, tracking is thought of as a Bayesian inference problem [29].

Filtering techniques in the tracking literature produces smoothed state estimates that is reasonable because objects tend to take smooth paths for energy efficiency. Among the filtering techniques, Kalman filter [27] is a popular choice for linear and Gaussian state and observation processes, however, it is not suitable for this setup as observations are generated through nonlinear camera projection processes. Extended Kalman filter [27] cannot handle highly nonlinear state and observation sequences [28]. Particle filters [30], on the other hand, can cope with nonlinear, nonGaussian and multi-modal state and observation processes [28]. Additionally, they are easy to manipulate, thus, suitable for adding other information, e.g., occlusion, when compared to other filtering techniques. More information for Bayesian filtering techniques and techniques that lead to the derivation of particle filter algorithm is provided in Appendices A, B, C.1, C.2, C.3 and C.4.

Object representation and object state determines the type of observations to be obtained from the sensors. Object representation and state representation are going to be detailed in the next two subsections. In addition, multi-view object tracking requires a means to define the relation between objects and their observations as well as the relation among observations. These are going to be defined in the subsections following those two subsections.

3.2.1 Representation of Tracked Objects

In this study, the extent of an object is compressed to a 3D point at its center of mass, as if its mass is uniformly distributed. The benefit of point object representation is utilizing epipolar geometry [76] between optical centers of cameras and the point object. The point observations with respect to this point object also resides on this geometry. This relation is general and does not require objects to reside on a ground plane, as assumed in most of the homography-based techniques. In addition, one observation is expected to lie on the epipolar line with respect to another observation. Epipolar geometry exists unless optical centers of cameras and the point object are collinear [76], which is quite rare. Therefore, airborne objects, seaborne objects and those on a rough terrain can also be tracked with this representation.

The point observation with respect to a point object is the center of its silhouette. The silhouettes are generated using background subtraction techniques [43] and silhouette centroid takes the place of projection of object's center of mass to the camera plane. Lemma 1 points out this relation.

Lemma 1. *Consider an object's mass concentrated at N identical unit masses in 3D Euclidean space. The centroid of these masses is* $M_c = \frac{1}{N}\sum_{i=1}^{N} M^i$. *Let* m^i *denote the projection of the* i^{th} *mass to an image plane. The centroids of these projections are* $m_c = \frac{1}{N}\sum_{i=1}^{N} m^i$. *The centroid of their projections are equal to the projection of their 3D centroids, if depth variation of these unit masses are negligible with respect to their distances to the projection center of the camera.*

Proof. Let $M^i = \begin{bmatrix} X^i & Y^i & Z^i \end{bmatrix}^T$ denote the 3D camera coordinates of the i^{th} unit mass. Its projection onto the image plane considering pinhole camera model [29, 76] is $m^i = \begin{bmatrix} u^i & v^i \end{bmatrix}^T$ where $u^i = X^i/Z^i$ and $v^i = Y^i/Z^i$. The centroid of these projections can be written as $m_c = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N}\frac{X^i}{Z^i} & \frac{1}{N}\sum_{i=1}^{N}\frac{Y^i}{Z^i} \end{bmatrix}^T$. The projection of their center of mass is $\tilde{m}_c = \begin{bmatrix} \frac{\frac{1}{N}\sum_{i=1}^{N} X^i}{\frac{1}{N}\sum_{i=1}^{N} Z^i} & \frac{\frac{1}{N}\sum_{i=1}^{N} Y^i}{\frac{1}{N}\sum_{i=1}^{N} Z^i} \end{bmatrix}^T$. Assume that there exists a Z^* such that $|\Delta Z^i| = |Z^i - Z^*| << |Z^*| \forall i$. This assumption states that the depth variation of the unit masses $|\Delta Z^i|$ are negligible with respect to their average distance to the projection center of the camera $|Z^*|$. Replacing Z^i with $Z^* + \Delta Z^i$ and neglecting the ΔZ^i, the centroid of projections and projection of the centroid becomes $m_c = \begin{bmatrix} \frac{1}{N}\frac{\sum_{i=1}^{N} X^i}{Z^*} & \frac{1}{N}\frac{\sum_{i=1}^{N} Y^i}{Z^*} \end{bmatrix}^T$ and $\tilde{m}_c = \begin{bmatrix} \frac{\frac{1}{N}\sum_{i=1}^{N} X^i}{Z^*} & \frac{\frac{1}{N}\sum_{i=1}^{N} Y^i}{Z^*} \end{bmatrix}^T$, respectively. Clearly, $m_c = \tilde{m}_c$, if depth variation of the unit masses are negligible with respect to their distances to the projection center of the camera, *i.e.*, $|\Delta Z^i| << |Z^*|$. □

The importance of this lemma lies in the fact that based on the depth range of the object and distance between object and the camera, one can assume safely the equivalence of projection of the object center of mass and silhouette centroid.

3.2.2 Object State Representation

Object state might contain all the useful and observable information about the object itself, other moving objects, sensors and the environment. The information about the object can be

typically listed as position, velocity, pose, dimensions and appearance. Object pose cannot be observed reliably, so it is excluded from the state list in this study. Other moving objects might occlude the object causing an increase in observation uncertainty, which in turn influence object state estimation. In addition, they might block its path [78, 79] resulting in a maneuver. Therefore, position, velocity and dimensions of other objects should be included into the object state. Sensor calibration, synchronization and field of view act indirectly on the state through acting on observations. Environment is another actor of object state, as it can be composed of structures, such as roads, lanes, sidewalks, bridges, doors, obstacles and static occluders. Although incorporation of additional information from the environment is expected to elevate tracking performance [58], such an approach is postponed to future studies and not implemented in this thesis work. Within this context, we consider position, velocity, dimensions and appearance of the object as object state, as well as positions and dimensions of other moving objects. A requirement to track all objects arises as any object might occlude any other during the course of its motion. These occlusions might be complicated involving multiple objects. Hence, occlusion interrelationships should be separated from object trackers and handled by an occlusion filter. Occlusion filter cooperates with object trackers and they together accomplish tracking multiple objects by taking *occlusion* into consideration (See Figure 3.1). Occlusion filter takes position and dimension states of moving occluders to compute probability that objects are being occluded. These probabilities are fed back to the individual object trackers. By introducing the occlusion filter, the complicated problem of tracking multiple objects by taking their occlusion interrelationships is simplified. As a result, states of object trackers contain position, velocity, dimensions and appearance. The states of occlusion filter are occlusion indicator functions of all tracked objects, which are binary random variables.

The positions and velocities of point objects constitute their position and velocity states. Although objects are represented by points, they are extended objects [80] in reality. Their extents contain information on the degree of occlusion objects might be experiencing. For this reason, object's silhouette size, *i.e.*, height and width, in each camera view constitute the dimension state. In addition, appearances of objects in each view are also included into the object state in order to increase accuracy of associating silhouettes to object trackers. The appearance is selected as the color histogram of the pixels within the silhouette, so that nonrigid objects and objects experiencing sharp pose change can still be related in time. So, color histograms of silhouettes in each view constitute the appearance state. Therefore, the object state can be written as $s = \{P, V, D, A\}$ where capital letters stand for position, velocity, dimension and appearance, respectively.

3.2.3 Relation Between State and Observations

Objects are represented by points and corresponding observations are defined as silhouette centroids by the help of Lemma 1. In the ideal case, lines emanating from the point object in 3D space to the optical centers of cameras pass from centroids of object's silhouettes. If line joining optical centers of two cameras are drawn, then a plane is formed by three noncollinear points; point object and optical centers of two cameras. This plane is called *epipolar plane* [76]. The intersection of epipolar plane with image planes of cameras is a line called *epipolar line*. Silhouette centroids in image plane of two cameras, which are on the epipolar plane, satisfy the epipolar constraint, which can be written as,

$$u_2^T F u_1 = 0, \tag{3.1}$$

where u_1 is homogeneous coordinate of the image of point object in camera 1 and F is the 3-by-3 Fundamental matrix [76]. Epipolar line, l, with respect to image point u_1 is defined as,

$$l = Fu_1. \tag{3.2}$$

Therefore, if Equation 3.2 is replaced in Equation 3.1, then epipolar line in camera 2 crosses image of point object in camera 2. To look for image of point object in a camera view given its image in another view, given image is multiplied by fundamental matrix to obtain epipolar line in the other view. Candidate images in the other view are checked whether they are on the epipolar line. Point objects and their point images in multiple views can be related by epipolar planes between any camera pairs.

3.3 Problem Setup

We consider a general scenario containing two stationary cameras with overlapping fields of view and multiple moving objects (*e.g.*, people). Our goal is to estimate the 3D coordinates of *all* objects in the scene recursively over time using the information gathered from the cameras, which we will refer to as *tracking* from this point on.

The assumption of two cameras is for the sake of illustration; however, all the following derivations can readily be extended to any number of cameras. For consistency, two cameras are used in the experimental results (Section 3.8). The accuracy of tracking is expected to increase for a larger number of cameras [81]. Moreover, the cameras are not assumed to be parallel stereo pairs. In fact it is even better to place them with their viewing directions orthogonal to each other, for better localization performance [54]; however, we assume no such arrangement in this thesis work. The cameras are assumed only to be fixed and their calibration information is assumed to be available.

We assume that the cameras are synchronized. If this assumption is not valid, the observation from each camera can be fused separately, as they become available at the correction stage of the tracker filter (Section 3.5.2.2). In this case, a modification to computation of the "epipolar mismatch" term in the measurement association (Section 3.4.4) could also be required, but this is not necessarily to be taken care of. Thus, for the sake of illustration, we continue by the synchronized measurements assumption.

Every frame captured at each camera is first processed and measurements (observations) per moving object per camera are generated. The details of the measurement models are explained in Section 3.4. Then, these observations are fed into particle filter trackers. We utilize one tracker per object in our framework (See Figure 3.1). The tracker filter is explained in Section 3.5. In this figure, we show there are N objects; however, the number N is not fixed, nor it is known a-priori: a filter is initiated as soon as a new object is detected in the scene and the filter of an object is destroyed once it leaves the scene. The filter initiation is explained in Section 3.5.3. The estimated positions are used in an *occlusion filter* (Section 3.6) to compute the conditional occlusion probabilities, given the estimated object positions. The trackers in turn utilize the conditional occlusion probabilities supplied by the occlusion filter.

We propose an occlusion-aware 3D particle filter tracking method that predicts dynamic occlusions and compensates the growing uncertainty by increasing the observation error accordingly. The increased error helps cope with occlusions by allowing good particle hypotheses to

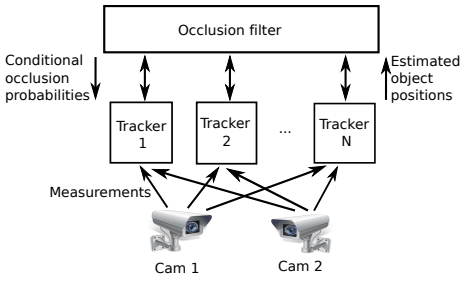


Figure 3.1: The proposed tracking framework.

survive. As a result of utilizing filtering techniques (mentioned as *tunneling* in [24]), tracking might continue even when target is occluded in one or both views. The advantages of 3D tracking, such as realistic object motion model, is also exploited in this work in such a way that the observations from two views are associated based on minimizing a cost function that is a linear combination of appearance mismatch, epipolar mismatch and prediction mismatch. Our contributions in this thesis study can be restated in detail as:

- A tracking structure composed of 3D particle filters per object together with a novel occlusion filter is formulated.
- By setting up the object position in 3D world coordinates, occlusion handling is simplified and occlusions involving multiple people is managed in linear order of computations.
- Observation error of the proposed filter is increased in accordance with the estimated occlusion probability so that uncertainty in object location during occlusion is taken into consideration.
- Increased observation error causes particles to expand due to iterative Monte Carlo simulations, that in turn widens the association gate so that location uncertainty is included in the association gate during occlusion, decreasing the chance to lose the object.
- Observation error is allowed to change by the dimension of the associated silhouette, so that distant and close objects can be tracked without parameter adjustment and particles are saved from degeneration when silhouettes are merged during occlusions.
- When objects come closer to each other for a long time, tracking is expected to continue despite their merged silhouettes, since it is allowed to assign an observation to multiple trackers. Tracking might continue, when they separate, since object appearances are utilized in associating silhouettes to trackers.

Next, the measurements from the cameras, and type of processing that is performed on the captured frames to obtain these measurements, are explained. Then, in the following sections, the object trackers and the occlusion filter are detailed.

3.4 Camera Measurements

From every frame captured at each camera, a set of measurements per object is generated. This set consists of a bounding box measurement, a point measurement, and an appearance measurement. The details on how these measurements are generated are presented in this section.

3.4.1 Silhouettes and Bounding Box Measurements

The first step applied to the captured images is to find the boundaries of the moving foreground objects in the image, namely "silhouettes". These masks are extracted through a Gaussian mixture background subtraction [43] algorithm. This is a per-pixel algorithm in which probability density of a pixel's intensity is approximated by a Gaussian mixture, that is learned and updated over time. Some tall and skinny (i.e., having larger weight and smaller variance) Gaussians in the mixture are labelled as background, since background pixels are more frequently observed. The other wider and smaller Gaussians in the mixture are labelled as foreground. The pixels in the acquired image are classified as foreground or background, depending on the label of the Gaussian with the highest probability density at the pixel's intensity. This labeling results in a foreground mask of moving objects. Connected image regions within the mask are segmented using connected component labeling methodology which results in silhouettes. Example silhouettes are shown in Figure 3.7 in Section 3.8.

Next, we define the bounding box of a silhouette to be the smallest rectangle containing it. The bounding boxes are utilized in various parts of our tracker; for example, in the Occlusion Filter (Section 3.6), hence, the height and width of the bounding boxes are regarded as measurements.

3.4.2 Point Measurements

As stated above, the goal of tracking is to estimate the 3D locations of all objects in the scene recursively over time. Since we assume connected and solid objects (such as people), it suffices to track one reference point for each object. In this paper, our goal is to track the *center of mass* of the objects.

One relevant measurement of the center of mass on a camera is the 2D coordinates of the projection of the center of mass on the image plane. For each object, we compute the centroid of the associated silhouette, and assume that it is a noisy measurement of the projection of the object's center of mass on the image plane. Although the projection of the center of mass is not exactly equal to the centroid of the silhouette, these two positions are close to each other if the depth variation of the object is small compared to its distance to the camera. This is a plausible assumption for our setup. This is more formally stated in Lemma 1.

3.4.3 Appearance Measurements

The appearance of each object on both camera views are also tracked in order to help with the measurement association step explained in Section 3.4.4. To quantify the appearance, we used the vectorized versions of the color histogram of the pixels within the associated silhouette, and denote it as appearance measurement.

3.4.4 Measurement Association

As explained above, a bounding box, a point and an appearance measurement are generated per silhouette on each frame from each camera. For correct tracking, correct association of these measurements to the objects (or equivalently, to the trackers) is needed. In other words, the generated silhouettes need to be associated to trackers.

For this purpose, for each tracker, we first determine the candidate silhouettes to consider for association in each view by thresholding their distance to an expected location on the image plane. Each tracker recursively estimates a probability density function (pdf) on the 3D object position (and other parameters such as velocity). In our tracker, these probabilistic distributions are recorded as weighted samples of the pdf, which are called "particles" (see Section 3.5 for details). These particles with 3D data are first projected on the 2D image plane. Then only those silhouettes, whose centroids' Mahalanobis distance to the projected particles are less than or equal to three standard deviations are considered for association. This is illustrated in Figure 3.2. In this figure, the white dots represent the silhouette centroids, whereas the blue dot represents the projected particles' mean and the blue circle represents the region within three standard deviations to the blue dot. Thus, those silhouettes with their centroids labelled with white dots are considered for association with this tracker, whereas the silhouettes corresponding to the other moving objects (such as the person with red jacket) are not considered.

This thresholding is performed on both views, and candidate silhouettes are determined. The cartesian product of the sets of candidate silhouettes from the two cameras produce candidate silhouette pairs. For each pair, a cost

$$J = \mu(M_A^1 + M_A^2) + \zeta(M_O^1 + M_O^2) + \gamma(M_E^1 + M_E^2) \tag{3.3}$$

is defined, where M_A is the appearance mismatch cost, M_O is the observation mismatch cost, and M_E is the epipolar mismatch cost. Details on the computation of these costs are given below. The superscripts signify the camera number, *i.e.*, M_A^1 is for view 1, and M_A^2 is for view 2. The coefficients in the cost computation are set to $\mu = 1$, and $\zeta = \gamma = 0.05$ for the whole experiments. The measurements from the silhouette pair with the minimum cost are passed on to the tracker as measurements. It should be noted that the silhouette association is not performed in a mutually exclusive manner; *i.e.*, one silhouette may be associated to more than one tracker. This approach is taken to account for the situation when objects are close to each other and their silhouettes merge; thus the measurements from the merged silhouette should go to the trackers of both of those close objects.

In the cost above, M_A is the appearance mismatch cost, which is the Bhattacharyya coefficient [82] between the tracked appearance (see Section 3.5) and that of the silhouette, subtracted from unity. M_O is the observation mismatch cost, which is simply the Euclidian dis-

Figure 3.2: Illustration of finding the set of candidate silhouettes to associate to an object's tracker. The white dots represent the silhouette centroids, whereas the blue dot represents the mean of the particles from the tracker, projected on the image plane. The blue circle represents the region within three units of Mahalanobis distance to the blue dot. Thus, those silhouettes with their centroids labelled with white dots are considered for association, whereas the silhouettes corresponding to the other moving objects (such as the person with red jacket) are not considered.

tance of the projected particles' mean to the silhouette centroid (the distance between the white dots and the blue dots in Figure 3.2). M_E is the epipolar mismatch, which is the distance of silhouette centroid to the epipolar line with respect to the centroid of the other silhouette in the pair. This is illustrated in Figure 3.3. In this figure, there are two silhouettes to consider at each view, which makes four pairs. The epipolar lines with respect to the centroids on the left view are drawn on the right view. The distances of the two centroids to the two lines, both on the right view, constitute the epipolar mismatch costs for the four pairs for the right view.

3.5 Tracker

In this section, the details of the tracker block in Figure 3.1 is given. Since each tracker tracks only one object and the data association is performed before tracking, the treatment in this section is as if there is only one object to track. However, there is a separate tracker for each object in the scene and all the objects are tracked by the proposed system. The tracker also utilizes the conditional probabilities of occlusion with respect to the other objects, which are supplied by the occlusion filter, and this information is assumed to be available in this section; the computation of these probabilities is explained in the next section.

Let $P_t \in \mathbb{R}^3$ be the position of the object's center of mass, $V_t \in \mathbb{R}^3$ be its velocity, $S_t \in \mathbb{R}^4$ be the heights and widths of the bounding boxes of the object's silhouette from both views and $A_t \in \mathbb{R}^{2k}$ be the color histograms of the object's silhouette from both views, where k is the histogram bin size. Note that these are the true values of these entities and they are unknown random processes. The subscript t in all variables denote time. The tracker's goal is to estimate P_t recursively over time given the measurements.

Figure 3.3: Illustration of the epipolar mismatch cost. There are two silhouettes to consider on each view, which makes four pairs. The epipolar lines with respect to the centroids on the left view are drawn on the right view. The distances of the two centroids to these two lines, both on the right view, constitute the epipolar mismatch costs for the four pairs for the right view.

A popular approach to this problem is using a Bayesian filter [83]. Among those, Kalman filter is not suitable for our setup, as observations are generated through nonlinear camera projection. Extended Kalman filter (EKF) cannot handle highly nonlinear state transition and observation mechanisms [28]. Additionally, nonlinear filters, such as EKF or unscented Kalman filter (UKF), are observed to fail handling discrete phenomena, such as occlusions [81], which is an important part of our work. Particle filters (PF) [30], on the other hand, can cope with non-linear, non-Gaussian and multi-modal state and observation processes [28]. For these reasons, we select particle filtering framework in our tracking approach.

Our tracker recursively estimates all of the entities listed above, namely P_t, V_t, S_t and A_t, concurrently, since our process model is so-called "constant velocity" model, which requires tracking of the velocity (Section 3.5.2.1), the bounding box sizes S_t are utilized in the occlusion filter (Section 3.6) and the "appearances" A_t in S_t are utilized in the data association step during the computation of the appearance mismatch cost (Section 3.4.4). However, noisy observations of the dimensions S_t and appearances A_t are directly available from the measurements (Section 3.4.1). This suggests that a simpler method, such as low-pass filtering of these measurements, might suffice for recursive estimation of S_t and A_t. It is also well-known that the number of particles needed for a successful PF implementation grows exponentially with the number of elements in the state (a.k.a. *"the curse of dimensionality"* [84]). In a time-critical application, such as surveillance, the available computational power limits the number of particles that one can use. Thus, in order to minimize the number of particles, while maintaining successful tracking, similar to the Rao-Blackwellization approach [85], we prefer to track the position P_t and velocity V_t with a particle filter, while the sizes S_t and appearances A_t are updated by a simple heuristic involving low-pass filtering of the relevant measurements. This filtering step is explained next.

3.5.1 Recursive Estimation of Bounding Box Sizes and Appearances

As explained in Section 3.4.1, the tracker is associated with two appearance measurements (one from each view) and two bounding box measurements each time step. These are noisy observations of A_t and S_t, respectively. Thus, the size S_t is estimated according to a unity gain first order auto-regressive low-pass filter

$$\hat{S}_t = (1-\beta)\hat{S}_{t-1} + \beta S_t^m, \tag{3.4}$$

where S_t^m denotes the bounding box size measurements at time t and $\hat{S}_t$ denotes the estimate of S_t. In Equation 3.4, β is selected to be equal to 0.1.

A similar update is performed on the estimate of appearances $\hat{A}_t$ only when the object is believed to be not occluded by any other object in the scene, since any occlusions will falsely alter $\hat{A}_t$ due to the appearances of occluding objects. Therefore, we first look at the occlusion probability of the tracked object based on the information provided from the occlusion filter (Section 3.6). If there is a non-zero probability of occlusion, $\hat{A}_t$ is *not updated*. Otherwise, in a similar fashion

$$\hat{A}_t^j = (1-\alpha)\hat{A}_{t-1}^j + \alpha A_t^m, \tag{3.5}$$

where $\hat{A}_t$ is the estimate of A_t, and A_t^m is the appearance measurements at time t. α is chosen to be 0.9, a value much larger than β. The reason for this is to react quickly to rapid changes in appearances due to pose and illumination changes.

In the next subsection, we will explain the details of particle filter tracker for the position and the velocity of the object.

3.5.2 Particle Filter Tracker for Position and Velocity

Let us define the particle filter (PF) state to be

$$x_t = \begin{bmatrix} P_t \\ V_t \end{bmatrix}. \tag{3.6}$$

The PF recursively updates an estimate of the posterior pdf of x_t, namely $p(x_t|z_{1:t})$, given all the measurements from the beginning to time t, which is denoted by $z_{1:t}$. The PF approximates this pdf by

$$p(x_t|z_{1:t}) \approx \sum_{i=1}^{L} w_t^{(i)} \delta(x_t - x_t^{(i)}), \tag{3.7}$$

where $x_t^{(i)}$, $i = 1, 2, \ldots, L$ are called the "particles", which are samples from the pdf $p(x_t|z_{1:t})$, $\delta(\cdot)$ is the Dirac-delta function, and $w_t^{(i)}$ are the weights of these particles. At any time, an estimate of the object state given the measurements can be computed by

$$\begin{aligned} \hat{x}_t &= \mathrm{E}(x_t|z_{1:t}) = \int x_t p(x_t|z_{1:t})\, dx_t \\ &\approx \int x_t \sum_{i=1}^{L} w_t^{(i)} \delta(x_t - x_t^{(i)}) dx_t = \sum_{i=1}^{L} w_t^{(i)} x_t^{(i)}, \end{aligned} \tag{3.8}$$

where E(·) denotes the expectation operator. Thus, the task of particle filter tracker becomes recursively updating the particles and their weights. This aim is achieved by the prediction and correction steps each time a new set of measurements arrive [1], which are explained next.

3.5.2.1 Prediction

We assume a constant velocity motion model [86]. Thus, particles are updated in the prediction stage according to

$$x_t^{(i)} = \begin{bmatrix} I & \Delta T\, I \\ 0 & I \end{bmatrix} x_{t-1}^{(i)} + \begin{bmatrix} \frac{(\Delta T)^2}{2} I \\ \Delta T\, I \end{bmatrix} \psi_t, \tag{3.9}$$

where ΔT is the time difference between two measurements and ψ_t models the acceleration of the target, which is assumed to be white Gaussian noise independent of all other processes, with a diagonal covariance matrix with standard deviations of 0.6, 0.6 and 0.1 m/s^2 in x, y and z directions, respectively.

3.5.2.2 Correction

First, we define an occlusion indicator function $\eta_t^{(j)}$, which is equal to 1, if the tracked object is visible in camera j at time t, and 0 otherwise:

$$\eta_t^{(j)} = \begin{cases} 1, & \text{object is visible in camera } j \text{ at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{3.10}$$

Then, the particle weights are updated in the correction step according to the Sampling Importance Resampling (SIR) particle filter algorithm [30]

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(z_t|x_t^{(i)}) = w_{t-1}^{(i)} \sum_{\eta_t^{(1)}=0}^{1} \sum_{\eta_t^{(2)}=0}^{1} p(z_t|\eta_t^{(1)}, \eta_t^{(2)}, x_t^{(i)}) \mathrm{P}(\eta_t^{(1)}, \eta_t^{(2)}|x_t^{(i)}). \tag{3.11}$$

Above, the term $\mathrm{P}(\eta_t^{(1)}, \eta_t^{(2)}|x_t^{(i)})$ is computed at the occlusion filter (Section 3.6). Assuming the observations from two cameras are independent given the particle state $x_t^{(i)}$ [81], the first term in the summation can be written as

$$p(z_t|\eta_t^{(1)}, \eta_t^{(2)}, x_t^{(i)}) = \prod_{j=1}^{2} p(z_t^{(j)}|\eta_t^{(j)}, x_t^{(i)}), \tag{3.12}$$

where $z_t^{(j)}$ denotes the measurement from j^{th} camera at time t.

The likelihood above is computed as follows: If it is conditioned on that the object is not visible at the camera's view (*i.e.*, $\eta_t^{(j)} = 0$), then this measurement is simply not used. This case results in a uniform pdf in the likelihood:

$$p(z_t^{(j)}|\eta_t^{(j)} = 0, x_t^{(i)}) = \frac{1}{\kappa}, \tag{3.13}$$

[1] Occasionally a resampling step is also needed to avoid particle deprivation, see [30] for details.

where κ is simply the difference between the maximum and minimum possible values of the measurement $z_t^{(j)}$.

On the other hand, if it is conditioned on that the object is visible at the camera's view (*i.e.*, $\eta_t^{(j)} = 1$) then a Gaussian pdf is used in the likelihood. First, particle i's 3D position is projected onto the camera j's view. Let this be $y_t^{(i,j)}$. Then

$$p(z_t^{(j)}|\eta_t^{(j)} = 1, x_t^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2(\eta_t^{(j)}, x_t^{(i)})}} \exp\left(-\frac{(z_t^{(j)} - y_t^{(i,j)})^2}{2\sigma^2(\eta_t^{(j)}, x_t^{(i)})}\right),$$

where

$$\sigma(\eta_t^{(j)}, x_t^{(i)}) = \frac{\sigma_t^{(j)}}{\mathrm{P}(\eta_t^{(j)} = 1|x_t^{(i)}) + \epsilon}. \tag{3.14}$$

Above, the occlusion probability $\mathrm{P}(\eta_t^{(j)} = 1|x_t^{(i)})$ is obtained from the occlusion filter (Section 3.6) and ϵ is set to 0.1 in order to bound $\sigma(\eta_t^{(j)}, x_t^{(i)})$ from growing unboundedly. $\sigma_t^{(j)}$ is equal to one third of the width component in the bounding box measurement S_t^m from camera j.

Equation (3.14) contains two important novelties. First, objects that have higher probability of being occluded are prone to noisier measurements. This is captured in Equation (3.14), since the measurement noise covariance is higher for such objects ($\mathrm{P}(\eta_t^{(j)} = 1|x_t^{(i)})$ is smaller). Moreover, objects that are closer to a camera might result in larger error in their centroid measurements compared to objects that are far away, since pose variations such as lifting a hand might result in large changes in bounding boxes for close objects. Equation (3.14) tweaks the measurement noise covariance automatically for this effect, since objects that are closer to the camera have larger bounding box widths $\sigma_t^{(j)}$. Additionally, if silhouettes merge, they also result in larger-than-normal bounding boxes. Since such point measurements are more erroneous, this situation is also automatically handled by Equation (3.14). We observed in our experiments that these ingredients improve the tracking accuracy considerably; therefore are among important contributions of this work.

As explained in the problem setup, the number of moving objects in the scene N is not assumed known or constant. Each time a new object is detected to enter the scene, a new filter is initiated for it, and discarded when the object leaves the scene. Next, we explain the filter initiation.

3.5.3 Initialization

When an object is detected to enter the scene, a new filter is created to track it. The detection is performed as follows: After the silhouette association step (Section 3.4.4), if there are any silhouettes left unmatched to any tracker, these are first tried to be matched to each other across the cameras. This is achieved by using a combination of the epipolar constraint and the proximity of the bases of the silhouette bounding boxes on the ground plane derived from the homography relation between the two views. If two bounding boxes from any two cameras are successfully matched for two consecutive frames, a moving object is declared to have entered the scene and a new filter is initiated for it. The appearances A_t and sizes S_t

Figure 3.4: Illustration of the occlusion probability estimation. The person with gray bounding box is occluding the one with light blue bounding box. The occlusion probability of the person closer to the camera is taken as zero, while that of the farther is computed as the ratio of the hatched area to the area of the light blue bounding box.

are simply initialized from the RGB color histograms and bounding box dimensions of the silhouettes, respectively.

The initialization of P_t and V_t involves creating particles for the PF, which means drawing random samples. We draw the initial samples of P_t from a multi-variate Gaussian distribution with mean equal to the triangulated position [76] of the silhouette centroids from both views, and a diagonal covariance matrix with standard deviations equal to 0.75 m in x and y dimensions and 0.3 m in z dimension. The samples of V_t are also similarly drawn from a Gaussian distribution with mean equal to the difference in triangulated 3D positions in two consecutive frames, divided into the time difference between two measurements. The covariance matrix for V_t samples is diagonal with standard deviations equal to 0.3 m/s in x and y dimensions and 0.15 m/s in z dimension. Above, the standard deviations in z dimension are always less than the others, since this dimension corresponds to height changes and they are expected to be less assuming a flat scene.

3.6 Occlusion Filter

The occlusion filter (see Figure 3.1) computes the conditional occlusion probabilities in Equations (3.11) and (3.14). Since all estimated posterior pdfs of the object states are available from the trackers, it is possible to compute the occlusion probabilities using these pdfs. However, this may become computationally prohibitive, if the number of objects in the scene is high [81]. One of the goals of this paper is to propose a practical multi-object tracker that also takes into account occlusions. Therefore, we will propose a computationally simpler heuristic approach to approximately compute the occlusion probabilities.

For this purpose, we first assume that the occlusion indicator function for camera-1 ($\eta_t^{(1)}$) and

that for camera-2 ($\eta_t^{(2)}$) are conditionally independent, given $x_t^{(i)}$,

$$\mathrm{P}(\eta_t^{(1)}, \eta_t^{(2)}|x_t^{(i)}) = \mathrm{P}(\eta_t^{(1)}|x_t^{(i)})\mathrm{P}(\eta_t^{(2)}|x_t^{(i)}). \tag{3.15}$$

Although this is not correct in general, especially for cameras that are placed close-by and have similar viewing directions [81], we made this assumption for computational simplicity and by considering the fact that usually limited number of cameras surveil a given location and they are distributed around the scene in different viewing directions.

The probabilities above must be computed for each particle i separately. However, for computational simplicity, for a given object, we prefer to use the estimated state and and use the same occlusion probabilities for all particles,

$$\mathrm{P}(\eta_t^{(j)}|x_t^{(i)}) \approx \mathrm{P}(\eta_t^{(j)}|\hat{x}_t). \tag{3.16}$$

The computation of the probability above is performed as follows (see Figure 3.4). The estimated position of the object $\hat{x}_t$ is first projected on camera-j's image plane. Let us call this $y_t^{(j)}$. The estimated bounding box of the object is positioned on the image plane with its centroid being at $y_t^{(j)}$. Let us call this the "object bounding box". Then the estimated positions of the other objects, which are obtained from other trackers, are considered. The objects that are estimated to be physically located between camera j's position and $\hat{x}_t$ are marked as "occluders". Their estimated bounding boxes are also positioned on camera j's image plane, in a similar fashion. Let us call these the "occluder bounding boxes". The ratio of the object bounding box covered by the occluder bounding boxes is assumed to be equal to $\mathrm{P}(\eta_t^{(j)} = 0|\hat{x}_t)$, and $\mathrm{P}(\eta_t^{(j)} = 1|\hat{x}_t) = 1 - \mathrm{P}(\eta_t^{(j)} = 0|\hat{x}_t)$.

It should be noted that the proposed occlusion filter is computationally fairly cheap. In addition to the simplifying assumptions, this is also made possible by the fact that we perform 3D tracking, as opposed to 2D. Therefore, depth-sorting the objects to identify the occluders becomes a straight-forward task. On the other hand, 2D trackers have less information on which object occludes which others; thus, end up generating likelihoods for all occlusion scenarios [8].

3.7 Analysis of Parameters of the Proposed Tracking Algorithm

The parameters used in this algorithm are adjusted to yield the best performance. The influence of these parameters to the performance is discussed next.

Learning rate of Gauss mixture background subtraction algorithm can cause detection errors, especially when scene objects remain stationary for extended periods of time. For this reason, learning rate should be made larger. However, setting it quite large prevents the algorithm to adapt to daily lighting variations. Typically, learning rate can be determined according to 1/(adaptation period x frame rate). If a pixel observes a color vector consistently for "adaptation period" long, then it is learned by the Gaussian mixture algorithm [42, 43]. For instance, setting the adaptation period to 80 seconds in a 25 frames per second video results in a learning rate of 0.0005.

Epipolar threshold is the parameter for the distance between the silhouette centroid and the epipolar line. Frame resolution and lens distortion basically determine this threshold value.

However, in those cases, where errors in silhouette centroids are larger due to fragmentation of silhouettes in cluttered scenes or shadows of the object, the threshold value should be made larger. On the other hand, this might cause irrelevant silhouettes to be associated across views. If the objects are distant to the cameras, this threshold should also be made smaller. On the contrary, it should be made larger, if objects are closer to the camera, as observation noise is expected to increase according to Lemma 1.

Observation noise is allowed to change by the variations in the silhouette's width so that the requirement of parameter adjustment due to changes in scale (i.e., objects moving away from the camera causes smaller silhouettes) is eliminated. However, smaller observation noise causes particles to become degenerate quickly, whereas, larger ones postpone particle's convergence to the object's state.

State transition noise is adjusted according to the amount of expected modeling error in the state transition model. State transition model in this study is constant velocity model. For this reason, if objects are maneuvering, then modeling error increases. Therefore, small state transition noise causes tracking failures, when objects are making turns. Large values, however, diminish the filtering (i.e., smoothing) ability of the filter such that resulting trajectories become rugged.

As particles sample the state space, number of particles determines the sampling efficiency. The dimension of the state determines the number of samples. For those problems, for which the state dimension is large, more particles are required. However, this number cannot be made infinitely large due to computational and storage constraints.

3.8 Experimental Results

The experiments are performed on a platform with Intel Core i5 CPU having 4 cores at 2.4GHz speed. The algorithm is implemented in an *unparallel* fashion for these experiments. Capacity of volatile memory (RAM) is 4GB and Ubuntu operating system is running on this platform.

During the experiments, proposed algorithm and *our implementation* of the algorithm in [14] are tested and their performances are compared using view 1 and view 2 of PETS 2009 benchmark dataset [68]. In addition, different versions of the proposed algorithm is generated according to occlusion-awareness strategies and their occlusion performances are compared. Beginning with the occlusion-unaware version in which observation noise variance is taken as a predefined constant, we define a 2D occlusion-aware version in which observation noise standard deviation is taken as one third of the width of the silhouette assigned to the tracker. In the 3D occlusion-aware version the observation noise in the 2D occlusion-aware version is divided by $1 - P_{occ}^{j}$ in view j so that occlusion causes more tolerance to those erroneous observations.

A total of 85 occlusions are counted from view 1 and view 2 of the PETS 2009 dataset [68]. Occlusion-unaware version failed tracking in 22 of them resulting in a 74% success rate. 2D occlusion-aware version yielded 14 failed tracking and a success rate of 84%. Finally, 3D occlusion-aware version failed in 13 occlusions resulting in a 85% success rate. In addition, 2D occlusion-aware version failed to provide sufficient observation uncertainty that causes a narrower gate after 3 of 85 occlusions; in 2 of which tracking failed, whereas 3D occlusion-

Table3.1: Comparison of success rates of different versions of the proposed algorithm. These versions possess varying levels of occlusion handling.

	Occlusion-Unaware	2D Occlusion-Aware	3D Occlusion-Aware
Success Rates	74%	84%	85%

aware algorithm managed all three occlusions. Performances of these versions are compared in Table 3.1. Therefore, 3D occlusion-aware version can also handle difficult cases where observation uncertainty is larger during occlusions.

Next, we compare 3D RMS tracking performance of our 3D occlusion-aware tracker to that of Khan and Shah [14], which is a well-known fusion based multi-view tracker. To compute the RMS accuracy, the ground-truth positions in 3D world coordinates of the objects are required which are not given in the PETS 2009 dataset [68]. For this purpose, we clicked on the feet positions of the tracked people manually in the video frames. Using the homography, targets' feet positions in world coordinate system are acquired. Only X and Y components of these positions are used as the ground-truth data.

Trajectories from our tracker and that of Khan and Shah are plotted in Figures 3.5 and 3.6 against their ground truth. Majority of the trajectories in Khan and Shah become fragmented as a result of frame rate of the dataset being lower than required. The linear tracks in Figure 3.5 are not fragmented and plotted for comparison. The average of the RMS errors of our algorithm for these trajectories is 0.15 meters, whereas that of our implementation of Khan and Shah's is 0.51 meters. The particular track in Figure 3.6 is selected, since this person experiences various occlusions during his nonlinear trajectory. The tracking still continues despite nonlinear motion and various occlusions in the proposed tracker. The resulting trajectory generated by Khan and Shah's algorithm is fragmented, when the person is passing behind the lighting pole in the left view. The reason of this fragmentation is that this person is occluded by the same other person in both views, as well as the pole in the left view.

Typically, Khan and Shah's algorithm produces false positives, when 2 cameras are used. Some of these false positives existed for more than 2 seconds depending on the relative motion of the people generating them. On the contrary, the proposed algorithm yielded no false positives. Moreover, Khan and Shah's algorithm lost occluded tracks that are close to their occluders in both views simultaneously. In contrast, the proposed algorithm manages these kinds of difficult occlusions, since it is aware of occlusions and benefits from object motion and appearance.

Furthermore, the proposed algorithm is able to handle occlusions involving multiple people as a result of tracking in 3D coordinates. The overhead of handling these difficult occlusions are not as much as those 2D algorithms, such as that proposed by Wu [8]. Unlike those 2D algorithms, where likelihoods for all combinations of "who is occluding who?" is computed using the observation. In 3D tracking, this information can be easily and reliably extracted by depth sorting the estimated tracker positions with respect to the camera centric coordinate frame. In Figure 3.7, the girl with red jacket is passing between 2 men in view 2 causing an occlusion scenario involving 3 people. In this figure, view 1 is shown on the top left and view 2 on the top right. Foreground masks of these views are at the bottom where top view of the common viewing area in the middle. Tracking is demonstrated to continue after the occlusion

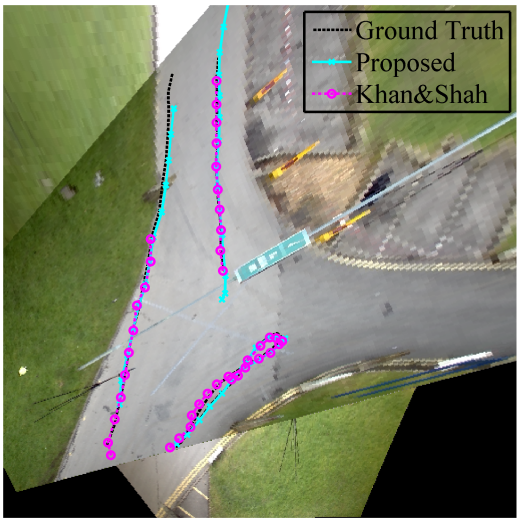


Figure 3.5: Comparison of proposed tracker with our implementation of Khan and Shah's [14] tracker on PETS 2009 dataset [68]. These people make linear motion.

in view 2 despite additional occlusions of other people in the scene. Notice that, particles of occluded trackers, in the right view of Figure 3.7, expand in the direction of epipolar line where the occluded silhouette is expected to exist. This expansion results in a gate alongside and containing the epipolar line, which enables the correct silhouette to be associated to the occluded tracker after the occlusion.

Additionally, we test the proposed algorithm using EPFL [88] and PETS 2006 [67] datasets. In the EPFL dataset [88], the selected cameras have about 135° of angle between them and they are placed horizontally. There is a person moving behind 3 people in one view and in front of them in the other view. Despite such an occlusion scenario, this person could be tracked by the proposed algorithm. The orange tracker in Figure 3.8 falsely followed after this occluded person. Despite this error, the proposed algorithm is experimented to work under different camera settings. The proposed algorithm also experimented in PETS 2006 dataset [67]. In this dataset, two of those cameras that produces less clutter are selected. These cameras have partial overlaps as can be seen in Figure 3.9. Left view is closer to the scene than the right view and this does not create a problem, since observation noise is a function of silhouette width. The proposed algorithm tracked multiple people despite their occlusions as illustrated in Figure 3.9. Therefore, the proposed algorithm can work in a camera setting where one camera is closer to the scene. The people could only be tracked in those regions where camera fields of views overlapped.

3.8.1 Execution Time Analysis of the Proposed Algorithm

The main loop of the proposed algorithm in which frame pairs are processed is divided into 8 sub-blocks, namely, frame grab, moving object segmentation, prediction, association, correction, tracker initialization, tracker candidate initialization and display. Execution times of

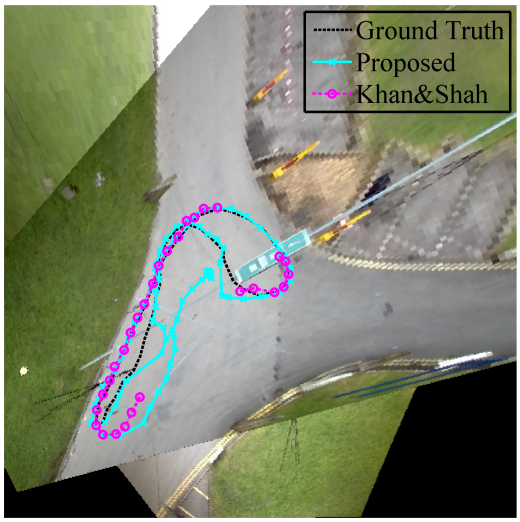


Figure 3.6: Comparison of proposed tracker with our implementation of Khan and Shah's [14] tracker on PETS 2009 dataset [68]. The person experiences occlusions throughout his non-linear motion.

these sub-blocks are computed, these times are presented in Table 3.2 by varying number of particles. Moving object segmentation sub-block took most of the processing time with average of 0.115 second as a result of its being a per-pixel algorithm. Reading two frames from computer memory took 0.38 second. Considering frame dimensions of $768x576$, the time spent seems reasonable. The execution times of prediction and correction sub-blocks decrease according to the decrease in the number of particles, since they repeat their procedures number of particle times.

Time analysis of the experiments with EPFL dataset is presented in Table 3.3. The frame dimensions of this dataset is $360x288$ which is almost half of that of PETS dataset in both dimensions. The number of particles during these experiments are set to 2000, so that the comparison is made with PETS 2009 dataset accordingly. The time taken for frame reading and moving object segmentation in EPFL dataset are almost one third of those of PETS 2009 dataset. The effect is the 4 times less number of pixels in EPFL dataset. The total execution time is 0.117 second in EPFL dataset, whereas it is 0.25 second in PETS 2009 dataset. The resulting frame rate on the average for EPFL dataset is 8.5 frames per second, on the contrary, average frame rate of PETS 2009 dataset is 4 frames per second, which is almost half of that of EPFL dataset. Therefore, reducing frame size pulls the processing times of frame pairs down.

In summary, reducing frame sizes improves the average frame rate of the tracker. In addition, creating multiple threads for those operations that can be processed in parallel, such as moving object segmentation, might improve the frame rate of the tracker.

Table3.2: Total execution time analysis table. One cycle of the algorithm is divided into sub-blocks where execution time of each sub-block is provided at various number of particles. Time unit is seconds.

	2000	1000	500	250
Frame Read	0.038	0.038	0.037	0.038
Moving Object Segm.	0.116	0.114	0.116	0.114
Prediction	0.007	0.004	0.002	0.001
Association	0.002	0.002	0.002	0.001
Correction	0.024	0.014	0.010	0.007
Initialization	0.031	0.017	0.011	0.006
Candidate Init.	0	0	0	0
Display	0	0	0	0
Total	0.25	0.22	0.21	0.20

Table3.3: Total execution time comparison between PETS 2009 and EPFL datasets. Time unit is seconds.

	EPFL	PETS
Frame Read	0.01	0.038
Moving Object Segm.	0.031	0.116
Prediction	0.004	0.007
Association	0.001	0.002
Correction	0.014	0.024
Initialization	0.018	0.031
Candidate Init.	0	0
Display	0	0
Total	0.117	0.25

Table3.4: Table of algorithm success rates. Proposed tracker is compared against color tracker and a combination of color and proposed tracker, where particle weights are computed both from geometry and color.

	Color	Proposed	Proposed+Color
Success rates	84%	85%	88%

3.8.2 Evaluation of Strategy of Weighting Particles by Color Similarity

We have modified the particle weighting equation of our algorithm to come up with a color tracker. The modification is that position likelihoods in two views are replaced by color likelihoods in particle weighting. Color likelihood is given in Equation 3.17 as,

$$f(z_t^a|x_t) = exp(Bhat(A_t, \hat{A}_t)) \tag{3.17}$$

$$Bhat(A_t, \hat{A}_t) = \sum_{k=1}^{K} sqrt(a_k \hat{a}_k) \tag{3.18}$$

where $\hat{A}_t$ is the color histogram of the image region corresponding to the projected particle position and A_t is the appearance state. This image region is determined by placing a bounding box centered at the projected particle position and is masked with foreground mask before histogram generation. In this tracker, any particle, whose appearance is the most similar to the tracker appearance, gets the largest weight. We tested this color tracker with PETS 2009 dataset. Projection of particles in this tracker became spread around the object image as a result of almost uniform color distribution on object images. The result of this spread is low localization accuracy, when compared to the proposed tracker. Different from the proposed tracker, color tracker managed to track objects wearing distinguishing colored clothes despite difficult occlusions. In addition, color tracker is able to track these objects when their silhouettes cannot be associated to the tracker. On the other hand, color tracker failed in these occlusion cases, for which objects involved in occlusion have similar colors unlike the proposed algorithm. We further combined the color tracker and the proposed tracker considering that the resulting tracker can have better occlusion performance. This combined tracker weights particles by multiplying both position (Equation 3.12) and color likelihoods (Equation 3.17). The resulting tracker succeeded in some occlusion cases where proposed tracker failed and failed in some of them where proposed tracker succeeded. Their occlusion performances are compared in Table 3.4. The color tracker produced a success rate of 84%, whereas proposed algorithm yielded 85% and the combined tracker has 88% of successful tracking. However, execution time of a single frame pair is 5.9 seconds for the color tracker, 0.25 second for the proposed tracker and is 5.7 seconds for the combined tracker. These considerably large execution times are a result of computing color likelihood, which involves extracting color histogram, for each particle.

In summary, the combined algorithm produced 3% increase in occlusion performance, while it took about 5.5 seconds more to process one frame pair on the average. Therefore, the performance can be improved by introducing additional likelihood terms at the cost of spending more time to process a frame pair.

Table3.5: Appearance model of the proposed algorithm is replaced with mean object color. This modified algorithm is compared against the proposed algorithm using RGB color histogram.

	Color Histogram	Mean Object Color
Success rates	85%	84%

Table3.6: Execution time table of the two algorithms. Time unit is seconds.

	Color Histogram	Mean Object Color
Execution Times	0.254	0.269

3.8.3 Mean Object Color Appearance Modeling

Instead of the RGB color histogram, a simple way of appearance modeling, mean object color, is also experimented. Mean object color, which is a vector of 3 elements, is defined as the ensemble average of color vectors from all pixels within the object silhouette. L2 norm of the distance between silhouette's mean color vector and appearance model of tracker is used to evaluate the similarity between them. Both occlusion performances and execution times of these algorithms are compared in Tables 3.5 and 3.6. The results indicate that the proposed algorithm produces better performance than the algorithm using the mean color. In addition, the execution time of the proposed algorithm is slightly less than this algorithm. In summary, using color histogram as appearance model produces better performance, while taking almost the same time when compared to the algorithm of which appearance model is mean object color.

3.8.4 Analysis of the Missing Observation from One View Case

When the tracker cannot take observations from one of the views, then the geometric constrain cannot be applied in the direction of this view. Particles can only be weighted by observations from the other view in this case. As a result, particles expand in the direction of the line joining center of mass of the object to the optical center of the camera where observation is available. This expansion is the result of simultaneous Monte Carlo simulations. The projection of this line to the view where observation is not available produces an elliptical spread of particles such that the long axis of the ellipse approximately coincides with the epipolar line. This is illustrated in Figures 3.10 and 3.11. The expansion of particles create an elliptical gate along the epipolar line where the silhouette centroid is expected lie. For this reason, when the object become observable in this views, the correct silhouette can be associated to the tracker and tracking can continue as before. Tracking continued in these figures.

3.8.5 Performance Comparisons with Social Tracking Algorithms

We compare our tracking accuracy to some recent trackers in the literature, namely [79] and [78]. We would like to point out upfront that the trackers of [79] and [78] use only one camera and they perform 2D tracking in the image coordinates, whereas we achieve 3D tracking in world coordinates using two cameras. Thus, it is not totally fair or straight forward to compare their results to ours. However, these works reported their CLEAR tracking accuracy metric [87], which assigns a percentage success score to the tracker. Pellegrini [78] is reported to achieve 64.5% score, whereas Leal-Taixe [79] is reported to achieve 67% score. Our proposed tracker achieves 81% accuracy in this metric.

3.9 Conclusions

We developed an occlusion-aware 3D multiple object tracker with two cameras and demonstrated its performance on the PETS 2009 data set. We have observed that inclusion of the occlusion filter in our framework increases the tracking performance, albeit to a limited degree. Our tracker is applicable to general tracking scenarios; objects closer to camera and farther away can be tracked without adjusting the parameters. Fusion-based algorithms yield false positives with only two cameras, whereas our algorithm resulted in no false positives. In addition, performance of the proposed algorithm does not depend largely on camera positioning unlike those algorithms that consider controlled environments.

The proposed algorithm can track objects with highly nonlinear trajectories that are experiencing multiple occlusions in both views. Even, our algorithm can handle occlusions involving 3-4 people. Moreover, our implementation of [14] failed to track targets passing behind a lighting pole, whereas the proposed algorithm managed this static occluder. The algorithm proposed in [13] is reported to fail to track people staying close to each other for a long time that is also observed in our implementation of [14]. In contrast, the proposed algorithm does not fail tracking in these cases as a silhouette can be assigned to multiple trackers in our association strategy. When these people get separated, their appearance models make them be assigned to the correct silhouettes.

Moreover, the proposed algorithm is able to work in different camera settings. According to our experiments with the EPFL dataset, the proposed algorithm is able to run with the horizontally located cameras such that the angle between cameras is larger than 90°. Additionally, the proposed algorithm is run with partially overlapping cameras in PETS 2006 dataset. Although one camera is closer to the scene than the other, the algorithm manages to track the objects.

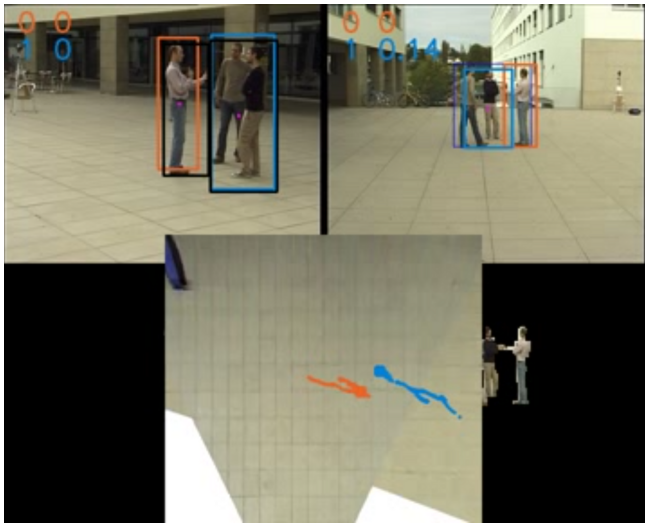
We have demonstrated that the proposed tracker can handle complex occlusion scenarios involving multiple people and it outperforms three other state-of-the-art trackers in the literature in terms of RMS or CLEAR tracking accuracy metrics.

(a) Before occlusion

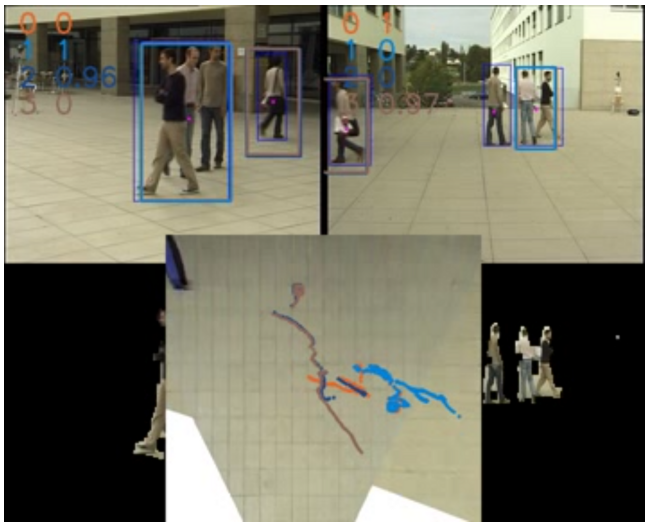
(b) During occlusion

(c) After occlusion

Figure 3.7: Illustration of occlusion involving 3 people on the right camera view. Notice in (c) that 2 men are occluding these 3 people after their occlusions are resolved. People in PETS 2009 dataset [68] are experiencing a series of occlusions, therefore it is more suitable to evaluate occlusion performance of the proposed algorithm when compared to PETS 2006 [67] and EPFL [88] datasets.

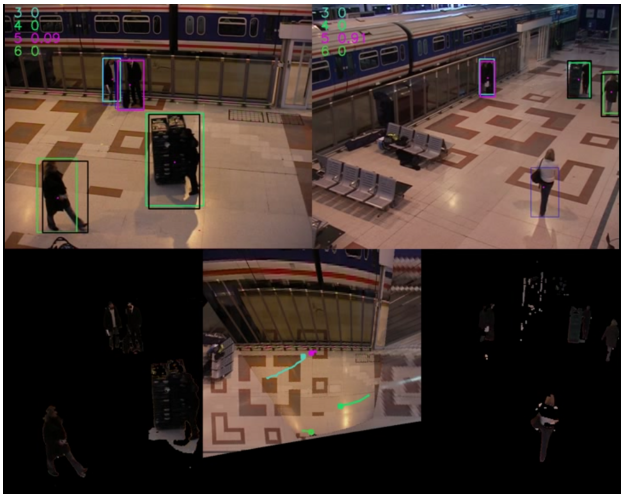


(a)

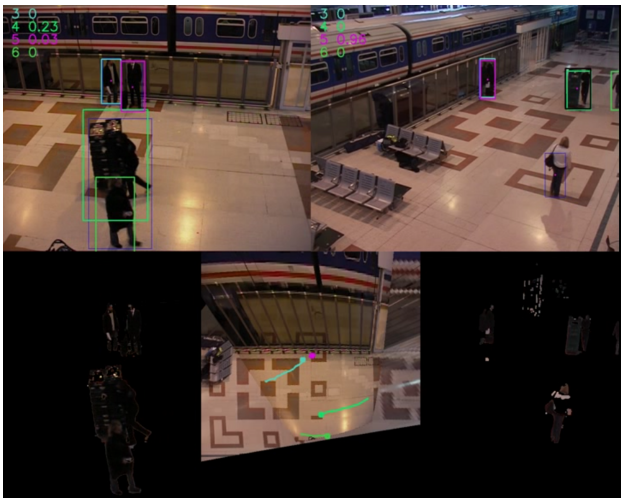


(b)

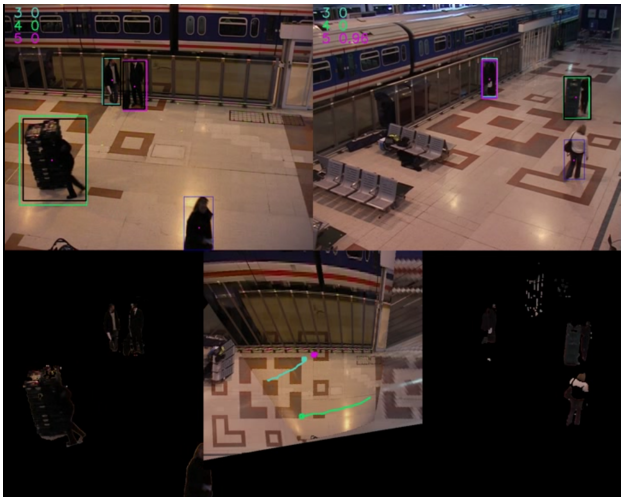
Figure 3.8: Illustration of tracking a person despite a significant occlusion. The person is tracked, despite he is being totally occluded in one camera during most of his motion.



(a)

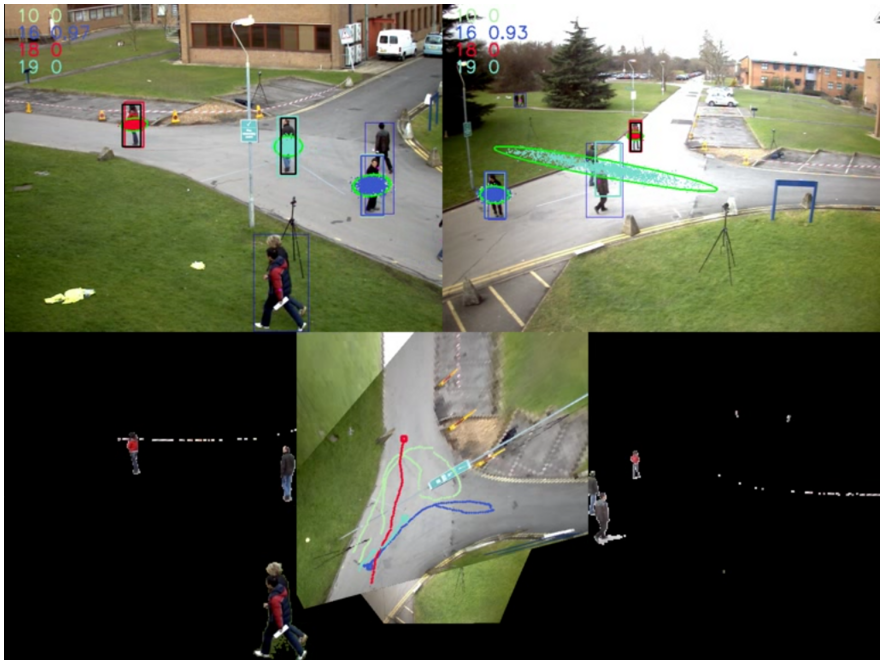


(b)



(c)

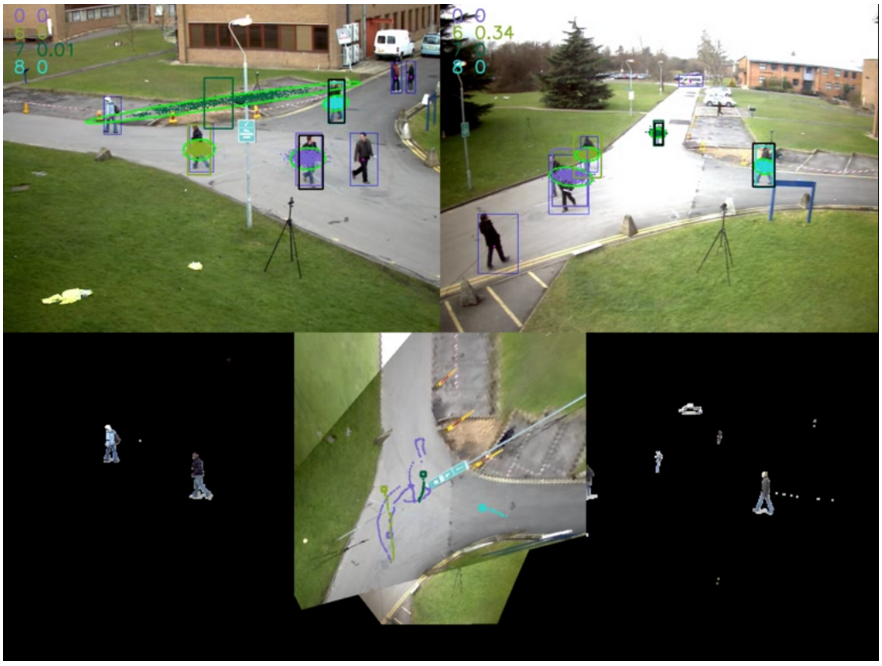
Figure 3.9: Illustration of tracking with PETS 2006 dataset [67]. People are tracked in a completely different environment and with a camera setting different from that in PETS 2009 dataset [68].



(a) Before

(b) After

Figure 3.10: Illustration of the case when observation from one of the views is not available. Particle weighting is performed with the likelihood only from the available view. The result is expansion of particles along the line joining object center of mass to the optical center of camera. The projection of this line to the unobservable view generates the epipolar line. Therefore, particles appear to be expanding along epipolar line in right view.



(a) Before

(b) After

Figure 3.11: Illustration of the case when observation from one of the views is not available. The same procedures occur as in Figure 3.10. This time unobservable view is the left one.

CHAPTER 4

SYNCHRONIZATION OF CAMERA NETWORKS

4.1 Motivation and Related Work

In Chapter 3, an occlusion-aware particle filter tracking algorithm is presented. Particles within a particle cloud are projected onto the image planes of the cameras. Particles weight themselves according to how well they fit in the observations. When the cameras are perfectly synchronized in time, the particle closest to the 3D target position gets the biggest weight. However, this is not the case when the cameras are not synchronized. A *position-shift* occurs between observations unless the target is stationary. As a result, particles cannot be weighted properly that causes degradation in tracking performance. In addition, when silhouettes are extracted from unsynchronized frames, then the scene point at the close proximity of lines emanating from silhouette centroids does not represent the true position of the object. A localization error occurs in these cases. For these reasons, this *position-shift* should be predicted and compensated.

The amount of position-shift between views due to lack of synchronization is closely related to the velocity of the object and the *time-shift* between frames. Therefore, position-shift differs from object to object according to its velocity. The velocities of objects can only be recovered by tracking them. The *time-shift*, on the other hand, is the same for all objects and depends on the capture times of frames in multiple views. More specifically, each camera has a local clock that can be modeled as [89],

$$C(t) = at + b \tag{4.1}$$

where a is the clock drift, which is close to unity, b is the offset of the camera clock and t denotes the *real* time. If two cameras are concerned, camera 1 and 2, the difference between their clocks can be written as [89],

$$C_1(t) = a_{12}C_2(t) + b_{12} \tag{4.2}$$

where a_{12} is the relative drift and b_{12} is the relative offset between clocks of two cameras. As camera operations depend on camera clock, the time difference between camera clocks create time-varying capture times of frames.

A solution to the problem of time difference between frame capture times of cameras is to synchronize them. Network Time Protocol (NTP) and Global Positioning System (GPS) are time synchronization schemes that increase cost and energy of the system [89]. NTP is currently used for synchronizing computers on the internet. However, it is not feasible for those cameras that are not connected to a computer as NTP requires extra energy and computational

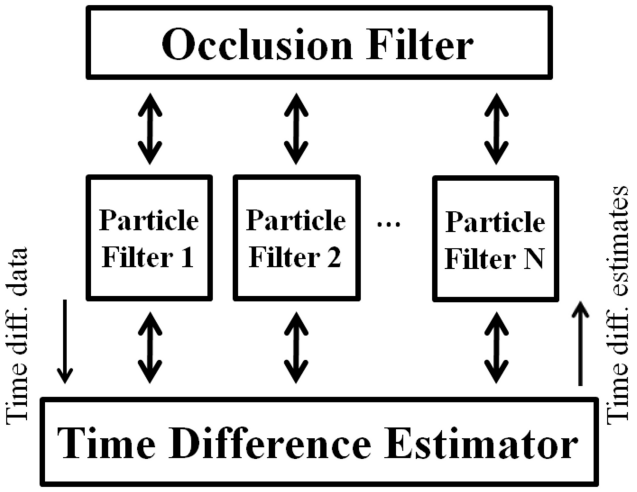


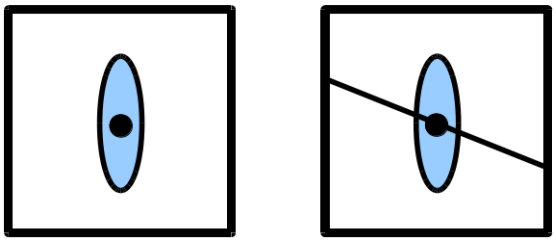
Figure 4.1: Occlusion-aware unsynchronized tracking system diagram. Occlusion filter keeps track of occlusion interrelationships of trackers. Time difference estimator smooths time difference computations of particle filters and produces its best estimate. The best estimate is fed back to particle filter trackers to correct the time synchronization error in their computations.

resources [89]. GPS, on the other hand, is expensive and cannot be available or reliable everywhere. Moreover, all network time synchronization methods operate by message transmission between nodes (computers or cameras in a network are called *nodes*). The amount of delay between the time transmitter send the synchronization code and the time receiver interpret the code causes a synchronization error [89]. Finally, synchronization of cameras by hardware requires skilled labor, and thus, increases the cost.

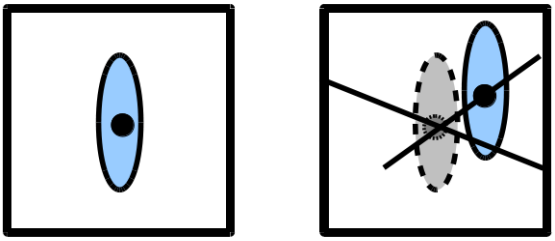
The time difference between frame capture times can be recovered using *timestamps* of the frames, i.e. the time a frame is captured, if the cameras are digital. However, analog cameras cannot produce frames with timestamps. In addition, as camera clocks drift relative to one another as in Equation 4.2, the timestamps could contain relative timing error in the long run.

Alternatively, algorithmic procedures for estimation of the difference between frame capture times can be developed. In the literature, there are a number of work on temporal synchronization. Stein [90] argues that time difference between views can be determined using trajectories of peoples and cars, once their spatial calibration is determined. Caspi [74, 91] approached this problem in two different ways. The first approach is to determine time difference between frames using sudden intensity changes in the video sequences. The second approach is to iteratively estimate spatial and temporal transformations using feature trajectories within a RANSAC-based algorithm. Whitehead [92] provided a neat definition of temporal synchronization terms and concepts. The author [92] looks for inflection points in trajectories where targets make a sharp turn and coarsely estimates the time difference. Later, epipolar line is utilized to come up with fine temporal synchronization that is sub-frame accurate. Stein [90] and Caspi's [74, 91] methods are iterative methods that require processing a number of frames before the estimation, and therefore, are not suitable for real-time applications. Whitehead's [92] algorithm cannot recover time difference, if targets do not make sufficient number of sharp turns.

In this thesis work, particle filter trackers explained in Chapter 3 tracks objects using obser-



(a)



(b)

Figure 4.2: (a) The epipolar constraint is satisfied in case of synchronized views. The silhouettes and their centroids are shown along with the epipolar line. (b) There is a position-shift when the views are not synchronized in time. The silhouette on the right view has moved during the time difference between views. The line along the motion of this silhouette, which is passing from silhouette centroids, is drawn together with the epipolar line in (b) on the right.

vations from unsynchronized view pairs. The system diagram of this method is provided in Figure 4.1. In Figure 4.1, N particle filters track objects within a scene, for which N is not known a priori. These particle filters give their positions and dimensions to the occlusion filter and receive their occlusion status. They also produce their estimates of time difference between frames and input these values to the time difference estimator. The estimator smooths these values and generate its best estimate of the time difference. The best estimate is fed back to particle filters where the effect of the position-shift is corrected.

4.2 Time Difference Filtering

Silhouette centroids in two camera views belonging to the same object satisfy the epipolar constraint, when the cameras are perfectly synchronized in time, as illustrated in Figure 4.2(a). However, epipolar constraint is no longer satisfied for non-stationary silhouettes, as in Figure 4.2(b), when cameras are not synchronized. This displacement is exploited to estimate the amount of time difference between unsynchronized views. The strategy is to estimate the displaced silhouette's velocity in image coordinates first, and then to estimate the distance taken by the silhouette during the time difference between views. The time difference is calculated by simply dividing magnitude of the displacement by magnitude of the velocity.

In practice, estimated values are spread over the time difference axis due to some error sources. If the object moves in 3D space such that its silhouette centroid remains on the

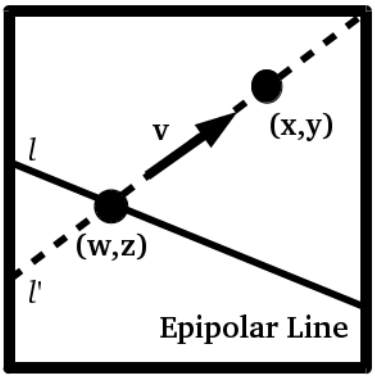


Figure 4.3: The estimation of the amount of time shift between views. This figure is generated by enlarging the silhouette motion illustrated in Figure 4.2(b) on the right. The dots on both sides of the velocity vector represent centroid of the same non-stationary silhouette at two time instants.

epipolar line, then the resulting values will be close to zero. The same happens, if the object remains stationary. This causes an increase in the frequency of observing values around zero which, in turn, causes estimation errors in the overall time difference value. In addition, when one of the silhouettes is merged with another occluding silhouette, the centroid of the resulting silhouette contribute to erroneous time difference estimations. For these reasons, stationary objects and those whose silhouette move along the epipolar line are detected and their estimations are filtered out. Estimates regarding silhouettes involved in an occlusion, i.e. either occluded or occluding, are filtered out as well. Furthermore, as the smaller angle between motion vector of a silhouette centroid and the epipolar line (Figure 4.2(b)) is getting smaller, small errors in the estimated silhouette velocity contributes to large errors in time difference estimation. These cases are detected and their estimates are filtered out also. Moreover, the estimates between the time interval, $[-3, 3]$ (in seconds), are considered for time difference estimations, since this range of values are the feasible ones. Another source of error is the result of imperfections in the silhouettes that shifts the image plane position of their centroids. Time difference estimator algorithm takes care of these errors.

4.2.1 Geometry of Time Difference Estimation

Figure 4.3 illustrates the geometry of the time difference estimation. The line *l'* that passes through the coordinate (x, y) is aligned with the silhouette's velocity vector and crosses the epipolar line l at the coordinate (w, z). The coordinate (x, y) is the location of the silhouette centroid and (w, z) is its location, if the views were synchronized. The epipolar line is computed using the epipolar constraint [76], $\alpha_2^T F \alpha_1 = 0$ where F is the 3-by-3 Fundamental matrix [76] between two views and α parameters are image plane coordinates in view 1 and 2. Coefficients of the epipolar line in view 2 result, when Fundamental matrix is multiplied by the image plane coordinate in view 1, $l = F\alpha_1$. The epipolar line equation is computed using the Fundamental matrix between views and the coordinate (x, y) is the first moment of the associated silhouette. The velocity vector, $v = \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T$, is computed by smoothing the dis-

placement of the silhouette centroid for the last 10 frames. The problem now is to determine the point (w, z) so that displacement of the silhouette, Δd, can be obtained. Suppose the equation of the epipolar line is in the form, $ax+by+c = 0$ and that of l' is $v_1(z-y)-v_2(w-x) = 0$. Intersecting the lines l and l' at (w,z),

$$aw + bz + c = -v_2(w - x) + v_1(z - y) \tag{4.3}$$

$$w = \frac{v_2x - (b - v_1)z - v_1y - c}{a + v_2} \tag{4.4}$$

is found. Replacing w in the epipolar line equation,

$$\frac{a(v_2x - v_1y - c)}{a + v_2} + \frac{a(v_1 - b)z}{a + v_2} + bz + c = 0 \tag{4.5}$$

$$z = \frac{-av_2x + av_1y - v_2c}{av_1 + bv_2} \tag{4.6}$$

z is determined. Therefore, the strategy is to compute z as in Equation 4.5 first and replace it in Equation 4.3 to calculate w. If displacement is denoted as $\Delta d = \begin{bmatrix} x - w & y - z \end{bmatrix}^T$, the time difference, Δt, is computed according to

$$\Delta t = \text{sgn}(\Delta d \cdot v)\frac{\| \Delta d \|}{\| v \|}. \tag{4.7}$$

where $sgn(.)$ is the sign function and $\cdot$ is the vector dot product. Here, the norm of the velocity vector *must* not be zero or a small value for practical reasons.

4.2.2 Incorporation of Time Difference into the Particle Filter Tracking Algorithm

The dynamic system equations of the particle filter trackers, which are detailed in Chapter 3, is presented in Equations 4.8 and 4.9. In Equation 4.8, x_t is a 6D vector containing position and velocity states in augmented form, ΔT is the inverse of frame rate and ψ_t models the acceleration of the target, which is assumed to be white Gaussian noise independent of all other processes. z_t is point observation, $h(.)$ is the nonlinear perspective projection function and v_t is observation noise in Equation 4.9.

$$x_t = \begin{bmatrix} I & \Delta T\, I \\ 0 & I \end{bmatrix} x_{t-1} + \begin{bmatrix} \frac{(\Delta T)^2}{2} I \\ \Delta T\, I \end{bmatrix} \psi_t, \tag{4.8}$$

$$z_t = h(x_t) + v_t \tag{4.9}$$

As stated earlier, one of the views is taken as reference and time difference is computed with respect to this view (let us call it "view 1" and call the other camera "view 2"). Accordingly, particle filter state is kept in accordance with this reference view. The particle filter trackers presented in this chapter are essentially the same as they are detailed in Chapter 3, except for the measures taken for time difference compensation between views. As a result of lack of synchronization between views, the state of the tracker at the time the frame of view 2 is captured is different from the state in Equation 4.8. Calling this state "$\widetilde{x_t}$", these two states are related as,

$$\widetilde{x_t} = \begin{bmatrix} I & \Delta t\, I \\ 0 & I \end{bmatrix} x_t \tag{4.10}$$

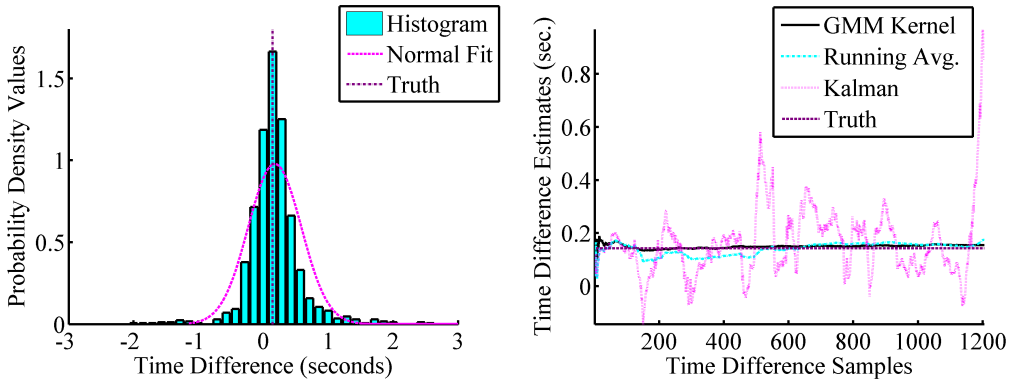


Figure 4.4: (a) The distribution of frame difference values. Those values whose magnitudes are larger than 10 are filtered out. The amount of frame difference is +1 (This frame difference is equal to $1/7 \cong 0.14$ second) for this experiment. (b) Comparison of cumulative running average method and Gaussian mixture kernel density estimation method. GMMKDE method are not affected by outliers as CRA method. Both methods end up close to the center of mass of the histogram in (a).

where tracker position is corrected whereas its velocity is remained unchanged in harmony with the constant velocity assumption. The point observations in view 2 should be evaluated based on this time-compensated state. Equation 4.9 is re-written as,

$$z_t^1 = h(x_t) + v_t^1 \tag{4.11}$$

$$z_t^2 = h(\widetilde{x_t}) + v_t^2 \tag{4.12}$$

where observation noise terms v_t^1 and v_t^2 are uncorrelated white Gaussian random processes. In the correction stage of the particle filter, when calculating particle weights, associating silhouettes to trackers and determining gates, and during the operations of the occlusion filter, the state in Equation 4.10 is utilized. Moreover, as silhouette pairs in unsynchronized views are not expected to satisfy the epipolar constraint, the association cost is computed as,

$$J = \mu(M_A^1 + M_A^2) + \zeta(M_O^1 + M_O^2) \tag{4.13}$$

without the cost from epipolar mismatch.

Similar changes should also be applied to the object detection, however, current detection strategy does not allow application of such changes. This change can be possible when candidate trackers are employed in the detection stage. In the current implementation and in these experiments, epipolar constraint is utilized in detection stage, where threshold for epipolar mismatch is increased to take the effect of unsynchronization into account. The effects of this detection strategy on the overall performance is negligible. However, employing candidate trackers in detection stage possess additional advantages, such as evaluating the temporal consistency of the detection; thus, decreasing the number of false detections.

4.3 Experimental Results

Experiments are conducted on the PETS 2009 [68] benchmark dataset. Frame difference between views is created artificially that is converted to time difference by dividing by the frame rate in the experiments. The frame rate of this dataset is about 7 frames per second. In these experiments, time-difference estimations, whose absolute value are larger than 3 seconds, are filtered out and not considered. The estimated time difference values (Equation 4.7) have quite a range, as can be seen in Figures 4.4(a), 4.6(a), 4.7(a), 4.8(a) and 4.9(a). These values need to be smoothed (i.e. filtered) as they appear. Three methods are experimented, namely, cumulative running average (CRA), Kalman filter [27] and Gaussian mixture kernel density estimation (GMMKDE) [43] methods. The CRA is the recursive version of the ensemble average operation. The equation of CRA is given as,

$$\Delta t_{n+1}^{CRA} = \frac{n}{n+1}\Delta t_n^{CRA} + \frac{1}{n+1}\Delta t_{obs} \tag{4.14}$$

where Δt_n^{CRA} denotes time difference estimate by CRA method at the n_{th} sample and Δt_{obs} is the observed estimate.

The GMMKDE is a Parzen window kernel density estimation method, where the kernel is a Gauss mixture composed of five Gaussians. The window of each Gaussian in the mixture is determined as box windows (which resembles spectrum of ideal bandpass filters) of width 0.57 seconds centered around the mean. The learning rate is selected as $r = 0.002$, so that the last 500 samples contribute to the current time difference estimate. The weights of Gaussians in the mixture are updated according to [43]

$$w_{n+1} = (1-r)w_n + r\mu_n \tag{4.15}$$

where w_n is the weight at sample number n and μ_n is 1 if the current sample matches with this Gaussian and 0 otherwise. The mean and standard deviation of Gaussians are updated, if a time difference observation falls into the window of the Gaussian (namely, observation matches the Gaussian) as [43],

$$m_{n+1} = m_n + (\frac{1-r}{c_m} + r)(\Delta t_{obs} - m_n) \tag{4.16}$$

$$std_{n+1} = std_n + (\frac{1-r}{c_m} + r)(\Delta t_{obs} - m_n) \tag{4.17}$$

where m_n is the mean at sample n, c_m is the number of samples matched to the Gaussian. A sample matches to a Gaussian, if it is closer than 2.5 times standard deviation to the mean and a sample can only match to a single Gaussian. The order of the tests for a match is from the Gaussian with highest weight to the one with lowest weight. The coefficient $(\frac{1-r}{c_m} + r)$ can be approximated by $\frac{1}{c_m}$ for small c_m and by r for large c_m, so that GMMKDE becomes similar to CRA for small number of samples, as it quickly converges to the true mean and becomes similar to a moving average filter for large number of samples. The estimate of GMMKDE is the maximum a posteriori probability estimate that is the mean of the Gaussian with the largest weight. Finally, Kalman filter is designed to run as a 1D constant parameter estimator.

During the experiments, tracking performance decreased, as the magnitude of time difference between views increased. Reliable tracking cannot be done with the PETS2009 dataset when the magnitude of time difference exceeds 1.29 seconds. Tracking failures are observed when

Figure 4.5: Illustration of the tracking error made when tracking a maneuvering object. The tracking is performed as described in this chapter. The frame difference is equal to 5 frames. The man dressed in black at the junction loses his track shown in light blue, since he started moving back in the right view and remains stationary in the left. Time difference compensation cannot work effectively with maneuvering objects, especially when time difference is large.

objects are maneuvering, especially making sharp turns (illustrated in Figure 4.5). The right view is 0.71 seconds ahead of the left view. The man dressed in black at the junction already starts to move backwards in the right view whereas he is stationary in the left. The tracker of this man, shown with light blue rectangle, fails tracking him at this point.

In the experiments, the frames of the second view in PETS2009 dataset are shifted by 1, -3, 5, -7 and 9 frames. Those three estimators are run simultaneously by the same time difference data. The histogram of the time difference values estimated by the trackers is generated. The size of histogram bins are 0.14 second. The range of histogram bins is nicely arranged (e.g, [0.07,0.21)). The histograms for different time difference values can be observed in Figures 4.4(a), 4.6(a), 4.7(a), 4.8(a) and 4.9(a). The estimates of the estimation methods are plotted for comparison with respect to samples. These estimates is illustrated in Figures 4.4(b), 4.6(b), 4.7(b), 4.8(b) and 4.9(b). In addition, root-mean-square (RMS) error made by the estimation methods are computed. The outcome of these experiments is to decide on the estimator with the best performance. RMS errors of estimators are compared in Table 4.1.

According to the experiments, GMMKDE estimator produced the least RMS error in the first 4 experiments; however, it could not complete its trend to the true time difference value in the fifth experiment by the given amount of data as can be seen in Figure 4.9(b). In this experiment, none of the other two estimators could also converge to the true value. The data of which distribution is shown in Figure 4.9(a) has 6 modes. For this reason, GMMKDE is a suitable method for parameter estimation out of this data when compared to CRA and Kalman

Table4.1: RMS errors of three estimators are compared. The rows contain time-difference values at which the experiments are made. The columns contain the estimators. The error values are in terms of seconds. The GMMKDE performs better than the other two, except for the last row.

Time Difference	Cumm. Running Avg.	GMMKDE	Kalman Filter
0.14 sec.	0.023	0.014	0.1364
−0.43 sec.	0.058	0.041	0.177
0.71 sec.	0.065	0.053	0.153
−1 sec.	0.107	0.1	0.18
1.29 sec.	0.576	1.019	0.529

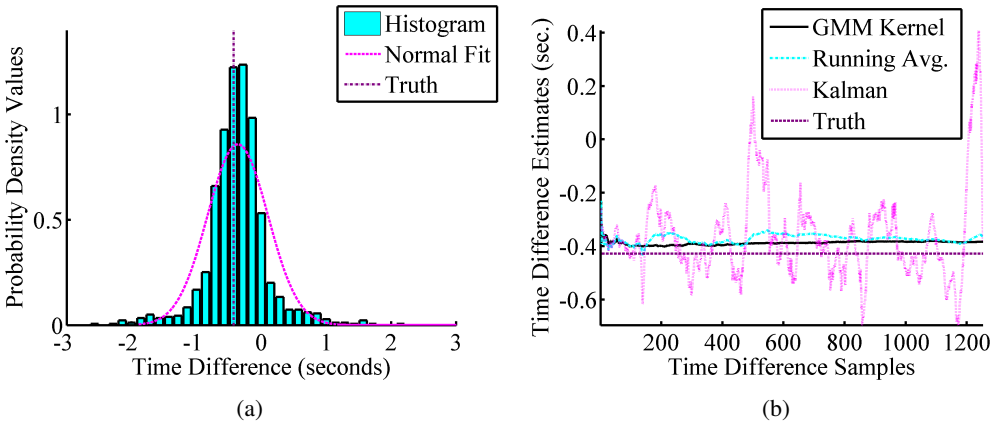


Figure 4.6: (a) The distribution of frame difference values. Those values whose magnitudes are larger than 10 are filtered out. The amount of frame difference is -3 for this experiment. (b) Comparison of cumulative running average method and Gaussian mixture kernel density estimation method. GMKDE method is trapped at a local mode at the beginning and recovers after sample 100. GMKDE ends up with a better estimate than CRA method.

filter. The sample mean of this data is close to 1 second, which is quite distant from the actual time difference, that is 1.29 seconds.

Kalman filter fluctuated around the true time difference value; therefore, yielded large RMS errors. CRA filter, on the other hand, performed close to GMMKDE. However, its infinite bandwidth, i.e. this estimator takes all samples into account no matter how distant they are to their estimates, resulted in small oscillations, on the contrary, estimates of GMMKDE was not affected by these erroneous samples.

In Figure 4.10, trajectory of a maneuvering person is illustrated. This person is tracked by a tracker detailed in Chapter 3, which does not take time difference into consideration in one experiment. In another experiment, He is tracked by a tracker proposed in this chapter, which takes time difference into account. Trajectory of this person obtained from both experiments are plotted on top of each other along with the true trajectory. RMS error of the trajectory, when time difference is not taken into account is 0.75 meters, whereas it is 0.58 meters, when time difference is taken into account. This clearly shows the importance of time difference

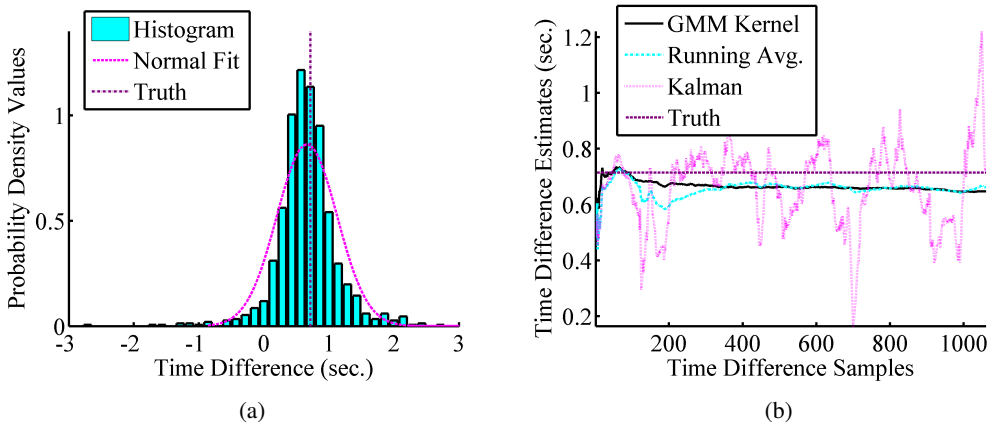


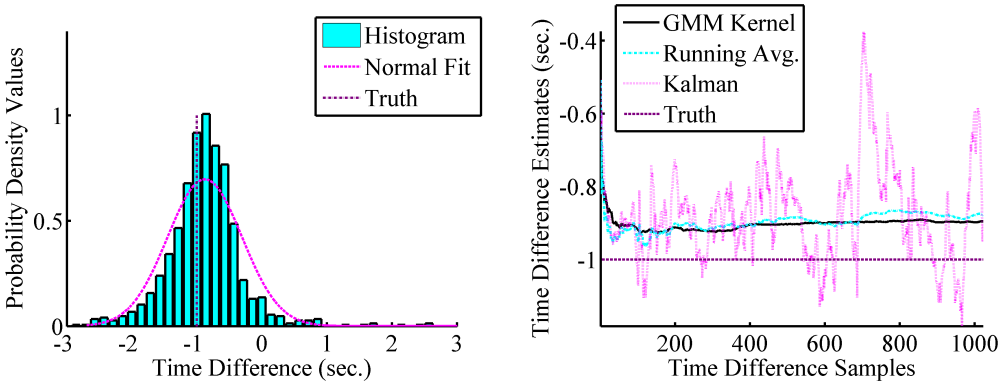
Figure 4.7: (a) The distribution of frame difference values. Those values whose magnitudes are larger than 10 are filtered out. The amount of frame difference is +5 for this experiment. (b) Comparison of cumulative running average method and Gaussian mixture kernel density estimation method. GMKDE method are not affected by outliers as CRA method. Both methods end up close to the center of mass of the histogram in (a).

compensation.

4.4 Conclusions and Future Work

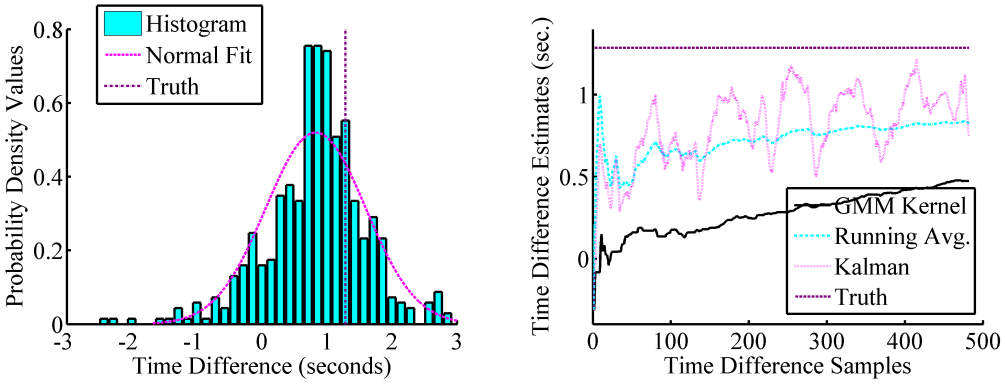
In this chapter, a novel method to estimate the time difference between multiple cameras is proposed and its performance is verified. The Gaussian mixture kernel density estimator (GMMKDE) is compared against cummulative running average estimator and Kalman filter and it performed the best during the experiments. GMMKDE produced estimates based on the time difference estimates of the trackers. These estimated values are used to compensate the motion of the target during this time difference. By this work, we show that it is possible to estimate the time difference during tracking targets. The proposed algorithm can estimate time differences up to 1 second. The performance of trackers degrades to unacceptable levels after 1 second that trackers cannot produce reliable time difference values. Our algorithm has advantages over those proposed in the literature. The first advantage is that estimation of time difference can be obtained while tracking targets. Secondly, different from those works that focus on planar scenes, our algorithm is able to work under general scenes without ground plane assumption. Moreover, our algorithm keeps track of time difference changes over time. Therefore, it is suited to those multi-view applications that are supposed to run for longer periods, such as months or years.

After the time difference between frames are estimated confidently, this difference should be corrected by taking necessary measures, such as delaying the camera that is ahead of the other. The time difference estimators should be reset and begin to estimate the new time difference. By iterating this procedure, the initial time difference can be reduced to bearable levels and possible time differences between views in the long run can be corrected.



(a) (b)

Figure 4.8: (a) The distribution of frame difference values. Those values whose magnitudes are larger than 10 are filtered out. The amount of frame difference is -7 (= −1 second) for this experiment. (b) Comparison of cumulative running average method and Gaussian mixture kernel density estimation method. GMKDE method is trapped at a local mode at the beginning and recovers after sample 100. GMKDE ends up with a better estimate than CRA method.



(a) (b)

Figure 4.9: (a) The distribution of frame difference values. Those values whose magnitudes are larger than 10 are filtered out. The amount of frame difference is +9 (≅ 1.29 seconds) for this experiment. (b) Comparison of cumulative running average method and Gaussian mixture kernel density estimation method. GMKDE method are not affected by outliers as CRA method. Both methods end up close to the center of mass of the histogram in (a).

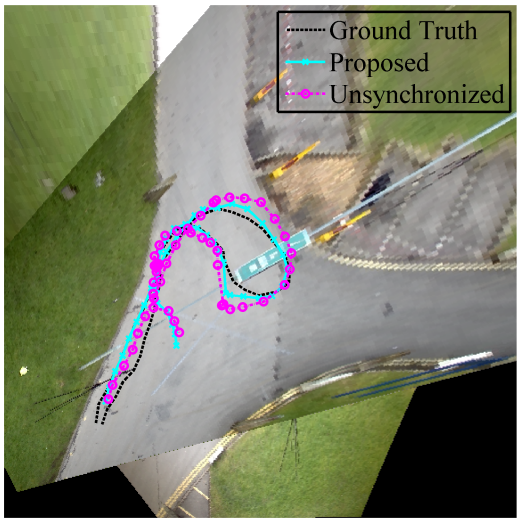


Figure 4.10: The trajectory of the same person is illustrated generated by a tracker detailed in Chapter 3 and a time difference compensated tracker explained in this chapter. There is a clear deviation of this person's position, when it is tracked by a tracker that does not take time difference into consideration. However, it can still track this maneuvering target, when the time difference between views is 0.71 seconds.

CHAPTER 5

RAO-BLACKWELL TECHNIQUE APPLIED ON THE PROPOSED TRACKING ALGORITHM

5.1 Motivation

Particle filters [30] simulate state evolution by Monte Carlo sampling as well as they represent posterior probability density of state sequence given a set of measurements. Particle filters can represent nonlinear, non-Gaussian and multi-modal state and observation processes, whereas Kalman filters [27] assume linear and Gaussian state and observation processes. Particles are the random samples from the state space. As the number of particles is increased, sampling efficiency increases, which in turn raise the accuracy of the estimate. However, number of samples cannot be increased further due to limitations on the computational resources. This limitation causes a problem, especially in those cases where state is nonlinear and high-dimensional (e.g. 27 dimensional state in [93]). In order to operate properly under these conditions, smart strategies for reducing the state dimension is applied. All these strategies can be named as "Rao-Blackwellization", if the resulting state estimator has lower variance than the estimator at the beginning. These strategies can be state decomposition, model simplification and data augmentation [94]. If some state components are conditionally linear on other components, then this conditional linearity can be exploited and these components are estimated by using Kalman filter. The remaining nonlinear components are estimated by particle filter as before. This analytical computation of state components is called marginalization, which is falsely used in place of Rao-Blackwellization in some published work. The whole idea is to come up with a new estimator with a minimum variance that is a sufficient statistics for the state. Rao-Blackwell theorem is the basis for this strategy [94].

In this thesis work, objects within the overlapping fields of views of two cameras are tracked in 3D world coordinates using particle filters by exploiting the epipolar geometry. Epipolar geometry is the projective geometry between two views that is only dependent on internal parameters of the cameras and their relative pose [76]. Representing objects as "point objects" allowed tracking them based on epipolar geometry. Silhouette's centers of mass in each view separately are compared against the projections of particle positions on these views to evaluate particles. As being point objects, object state has position and velocity components defined in 3D world coordinates. In addition, dimension of objects in each view is added to the state to estimate occlusion relationships between moving objects. This dimension is observed from the dimensions of bounding box including the silhouette. Object appearance in each view is also incorporated into the state and helps in data association. Therefore, object state is composed of 3D position, 3D velocity, bounding box dimension for each view and appearance

for each view. Position and velocity components are estimated by using a particle filter, while dimension and appearance states are smoothed by observed quantities in a linear convex combination relation. 6D state of particle filters is marginalized into 3D position state by estimating 3D velocity with a Kalman filter. Since each particle has these states, a Kalman filter estimates particle velocity for all particles. The estimate of Kalman filter is incorporated into the state transition relation of particle position, while position difference in consecutive frames divided by time duration between frames is input to the Kalman filter as pseudo-observation. In the previous particle filter algorithm, state transition noise was acting freely on both velocity and position. In the proposed Rao-Blackwellized algorithm, noisy position estimates act on particle velocities through the measurement update relations of the Kalman filter.

Although this algorithm should satisfy the Rao-Blackwell theorem to be called "Rao-Blackwellized", we will call it so naively without checking if it satisfies the theorem. Still, it can be argued that the proposed technique utilizes the main idea behind the Rao-Blackwell theorem.

5.2 Literature Review

Schon [93] divided 27 dimensional nonlinear state space into conditionally linear and nonlinear states. The linear part become linear, when conditioned on the nonlinear state. The linear state is "marginalized" using Kalman filter, while nonlinear part is filtered by a particle filter.

Kim [95] divided appearance state from position and velocity states in a Rao-Blackwellized manner for a single view visual tracking application. Appearance is taken as the orientation histogram and approximated by a Gaussian mixture. Each particle has a position and velocity state and appearance state corresponding to the 2D position. Appearance template is divided into regular subregions and Gauss mixture is extracted for each subregion to cope with partial occlusions. Observations are computed by using mean-shift algorithm.

Different from Schon [93], the state filtered by particle filters is 6 dimensional in this thesis work. In addition, appearance is not filtered by particle filters, thus is not a state of particles, as opposed to Kim [95]. Appearance templates are utilized in Kim's [95] method, therefore it is not expected to perform effectively, especially when objects are nonrigid and changing their pose.

5.3 State Decomposition

The state components of particle filters in Chapter 3, i.e. position and velocity states, can be written as,

$$x_t = \begin{bmatrix} P_t \\ V_t \end{bmatrix}. \tag{5.1}$$

where P_t stands for position at time t and V_t denotes velocity. The dynamic state equation can be expressed in an open form as,

$$\begin{bmatrix} P_t \\ V_t \end{bmatrix} = \begin{bmatrix} I_3 & \Delta T\ I_3 \\ 0_3 & I_3 \end{bmatrix} \begin{bmatrix} P_{t-1} \\ V_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{(\Delta T)^2}{2} I_3 \\ \Delta T\ I_3 \end{bmatrix} \psi_t \tag{5.2}$$

where ΔT is the time difference between consecutive frames in seconds, I_3 is 3-by-3 identity matrix, 0_3 is 3-by-3 zero matrix and random vector ψ_t models the acceleration of the object and motion modeling error, which is assumed to be originating from a white Gaussian process and independent of all other processes.

State is decomposed into linear and nonlinear components; velocity being the linear state and position is the nonlinear state. Linear state is analytically determined by a Kalman filter, whereas the nonlinear state is filtered by a particle filter in a Rao-Blackwellized fashion. These two filters should communicate with each other, since velocity is the time derivative of position and next position is interpolated using velocity. Since each particle has own position and velocity states, a Kalman filter per particle is going to estimate its velocity. State transition relation can be manipulated if Equation 5.2 is written in an open form,

$$P_t = P_{t-1} + \Delta T \; V_{t-1} + \frac{(\Delta T)^2}{2} \; \psi_t^1 \tag{5.3}$$

$$V_t = V_{t-1} + \Delta T \; \psi_t^2 \tag{5.4}$$

where ψ_t^1 and ψ_t^2 are identical to ψ_t and written as such, since position and velocity are states of different filters. If velocity estimate of Kalman filter, $\hat{V}_{t-1}$, is added and subtracted in Equation 5.3, Equation 5.5 is obtained.

$$P_t = P_{t-1} + \Delta T \; V_{t-1} + \Delta T \; \hat{V}_{t-1} - \Delta T \; \hat{V}_{t-1} + \frac{(\Delta T)^2}{2} \; \psi_t^1 \tag{5.5}$$

Equation 5.5 can further be manipulated as,

$$P_t = P_{t-1} + \Delta T \; \hat{V}_{t-1} + \Delta T \; \alpha_t + \frac{(\Delta T)^2}{2} \; \psi_t^1 \tag{5.6}$$

where $\alpha_t \triangleq V_{t-1} - \hat{V}_{t-1}$ is defined as the estimation error of the Kalman filter that is to be incorporated into the particle filter as additive noise component. The error term α_t is assumed to be zero mean Gaussian random vector with a covariance matrix, Q_{α_t}, which is equal to the a priori state estimation covariance matrix of the Kalman filter. Time dependence of the error term α_t is denoted by the subscript t. The difference between current and previous positions divided by the time duration between two consecutive frames, which is defined as $y_t \triangleq \frac{P_t - P_{t-1}}{\Delta T}$, can be used as a pseudo-measurement for the Kalman filter. Although this is not a real measurement from a sensing mechanism, giving noisy position difference to the Kalman filter as measurement smoothes out those noise terms. Therefore, particle velocities are affected from the *randomness* of the noise term less than the original particle filter. By selection of the velocity measurement as such, Kalman prediction should be implemented first, whose a priori velocity estimate is used in extrapolation of the next position. After generating the position estimate, measurement of the Kalman filter becomes available and Kalman correction can be implemented. The resulting state space equations of the Kalman filter is given in Equations 5.7 and 5.8.

$$V_t = V_{t-1} + \Delta T \; \psi_t^2 \tag{5.7}$$

$$y_t = \frac{P_t - P_{t-1}}{\Delta T} = V_{t-1} + \frac{\Delta T}{2} \psi_t^2 \tag{5.8}$$

On the other hand, the state space equations of the particle filter become,

$$P_t = P_{t-1} + \Delta T \; \hat{V}_{t-1} + \Delta T \; \beta_t \tag{5.9}$$

Table5.1: Performance table. The columns stand for the number of particles utilized in the experimenting by particle filters. The rows are the two algorithms of which performances are compared. The algorithm called "original" is the algorithm presented in Chapter 3. "RaoBlack" is short for the Rao-Blackwellized algorithm detailed in this chapter. The success percentages of the algorithms over number of particles are organized in this table.

Number of Particles	2000	1000	500	250
Original	85%	85%	80%	79%
RaoBlack	86%	86%	82%	81%

Table5.2: Total execution time analysis table. The columns stand for number of particles and rows show the algorithms. The total execution time is the total time spent when processing a frame pair. The execution time variation of algorithms over number of particles is clarified.

Number of Particles	2000	1000	500	250
Original	0.25	0.22	0.21	0.20
RaoBlack	0.46	0.34	0.26	0.23

$$z_t = h(P_t) + w_t \tag{5.10}$$

where $\beta_t \triangleq \alpha_t + \frac{\Delta T}{2}\psi_t^1$ is a zero-mean noise term with covariance $Q_\beta = Q_{\alpha_t} + \frac{(\Delta T)^2}{4}Q_{\psi^1}$, which is a result of the assumption that random vectors α_t and ψ_t are uncorrelated.

In summary, the resulting algorithm involves the following steps:

1. Predict the position (nonlinear) state for all particles by drawing a random sample from the random variable β and replacing it in Equation 5.9.
2. Update the velocity (linear) state with pseudo-measurement, which is given in Equation 5.8.
3. Predict the velocity state for all particles according to Kalman filter prediction equations [27].
4. Update the position states of all particles as described in Section 3.5.2.2.

The state of Kalman filter after the Kalman prediction is available as the most recent state at the Particle prediction and it is used in place of $\hat{V}_{t-1}$ in Equation 5.9. In addition, as the covariance matrix of the Kalman state after the Kalman prediction is available at this stage, predicted state covariance matrix of the Kalman filter is used for Q_{ψ^1} when drawing a random sample from β_t.

5.4 Experimental Results

During the experiments, the algorithm presented in Chapter 3, (named "original" in the tables), is compared against the Rao-Blackwellized algorithm using the PETS2009 benchmark

Table5.3: RMS errors of a nonlinear trajectory with respect to number of particles. The person is making a round turn while experiencing series of occlusions. This causes a modeling error in Rao-Blackwellized algorithm, since velocity is assumed to be linear in this algorithm. The unit of the errors is meter. (NA stands for not available)

Number of Particles	2000	1000	500	250
Original	0.35	0.36	NA	0.3
RaoBlack	0.37	0.36	0.41	0.38

dataset. The number of particles is decreased in both algorithms and occlusion performances are evaluated. Beginning from 2000 particles, the number of particles are decreased to 1000, 500 and 250. A total of 85 dynamic occlusions are counted in both views of the PETS2009 dataset. The number of tracking failures are counted in all the experiments for both algorithms. The counting is repeated to make sure that the results are correct. The performance is determined by the percentage of successful tracking during all occlusions in both views. The resulting performances are provided in Table 5.1. The performance of Rao-Blackwellized algorithm for all number of particles is slightly above those of the original algorithm. Rao-Blackwellized algorithm has 1% better performance than original algorithm when 2000 and 1000 particles are present. The difference is 2% in favor of Rao-Blackwellized algorithm when particle number is pulled down to 500 and 250. Therefore, Rao-Blackwellized algorithm has a slightly better occlusion performance than original algorithm, especially when number of particles are low.

The root-mean-square errors (RMS) of a nonlinear trajectory with respect to different number of particles are extracted in Table 5.3. This trajectory is shown in Figure 4.10. According to Table 5.3, RMS trajectory errors from both algorithms are close between 0.3 and 0.4 meter, even when the number of particles is 250. The error differences are in centimeter scale. Ground truth data, which is generated by clicking on the feet positions of the person, might not be accurate in the centimeter scale. Therefore, localization error is almost the same for both algorithms, although the person with nonlinear trajectory experienced a series of occlusions along this trajectory.

Next, execution times of both algorithms are evaluated to check, whether the reduction in the number of samples improved the rate at which frames are processed by the algorithm. The results are presented in Table 5.2. These execution times are computed for all frames and averaged. The execution time of the Rao-Blackwellized algorithm is always larger than the original algorithm for all particle numbers. This is the result of running a Kalman filter for each particle in the Rao-Blackwellized algorithm. The execution times of the original algorithm stayed below those of the Rao-Blackwellized algorithm.

5.5 Conclusion

In conclusion, 6D states of the particle filter trackers are divided into linear and nonlinear components such that velocity becomes the linear component and position becomes the nonlinear component. The velocity is filtered by a Kalman filter whereas position is filtered by a particle filter. As velocity is the time derivative of position and next position is extrapolated using

velocity, these two filters cooperated to accomplish the tracking. Monte Carlo samples of the particle filter sampled only 3D position space, while velocity is handled deterministically. The experimental results verified that the proposed Rao-Blackwellized algorithm produced the better performance than the original algorithm. The performance improvement by Rao-Blackwellized algorithm becomes more significant, when number of particles are decreased to 500 and 250. The localization accuracy is almost the same in both algorithms regardless of the number of particles. However, execution time of one frame pair in the Rao-Blackwellized algorithm is greater than that of the original algorithm as a Kalman filter is employed for each particle.

CHAPTER 6

RESULTS AND CONCLUSIONS

6.1 Summary

Surveillance problem involves inferring motives of people from their actions and these actions can be deduced from trajectories generated by those people. Tracking is a difficult problem, however, due to occlusion, illumination changes, etc. Especially, occlusion is a major problem of visual tracking that needs to be handled. Tracking in 3D world coordinates is preferable over tracking in 2D image coordinates, since reaching from trajectories to actions is easier and it is less viewpoint dependent in 3D. 3D tracking requires more than one camera and these cameras require to be synchronized in time. An occlusion-aware multiple-object tracking algorithm is proposed in this thesis such that an occlusion filter keeps track of inter-object occlusions and its performance is evaluated. Particle filter is used for tracking in this algorithm, as it can operate under nonlinear state and non-Gaussian noise processes. The observation noise of particle filters are varied according to silhouette widths so that variations in silhouette size, when object scale is changed and object become occluded are both taken into consideration. By manipulating observation likelihood terms, occlusion probabilities are incorporated into the particle correction stage of the particle filters, while these occlusion probabilities are computed by an occlusion filter. The proposed algorithm is also extended to work under unsynchronized cameras by compensating the motion of target within the time difference between cameras. A time difference estimation method is developed depending on the mismatch between epipolar line and silhouette centroids for this purpose. Each tracker passed their time difference computations to Gaussian mixture kernel density estimator that produces a maximum a posteriori estimate from those noisy computed values. Additionally, position and velocity states are divided as nonlinear and linear states, respectively, in a Rao-Blackwellized manner. Velocity state is marginalized using a Kalman filter while position state is filtered by a particle filter. The resulting Rao-Blackwellized tracker is experimented to work without performance degradation.

6.2 Conclusions

Occlusion handling improves tracking performance as it is verified in the proposed appearance based occlusion-aware particle filter tracking algorithm. Different versions of this algorithm is generated by varying the level of occlusion handling. The experiments showed that performance increased as the level of occlusion handling increased. It is also observed during the

experiments that particle filters is able to track objects with highly nonlinear state. Additionally, objects experiencing various occlusions, i.e., one-view, two-view occlusions, while they are maneuvering are experimented to be tracked. The algorithm performed well when objects are distant from the cameras, as well as when they are close to them. Moreover, weighting particles with an additional likelihood based on color similarity with the appearance model in addition to the position-based likelihood produced better performance; however, the average rate at which frames are processed dropped considerably. Furthermore, the proposed algorithm is compared with that proposed by Khan [14]. The accuracy of object trajectories generated by both algorithms are evaluated and proposed algorithm appeared to be more accurate than Khan's [14]. In addition, Khan's algorithm yielded many false alarms (a.k.a false positives) and failed when static occluders were present in the scene. Our algorithm produced no false positives and performed well under static occluders. Moreover, our algorithm is able to work under different frame rates and under different camera placement settings. The cameras are not required to be perfectly synchronized with the proposed algorithm. Additionally, the proposed algorithm was tested to work when observations from one of the cameras were not available. Tracking was observed to continue after these observations became available.

Analysis of execution times of various modules of the algorithm is achieved. The module that spends the most time is found out to be the moving object segmentation module. The average rate at which frames are processed are experimented to be pulled down by reducing the frame sizes. In addition, parallel implementation techniques can be applied to some of those modules to this average rate.

It is demonstrated that tracking targets in unsynchronized camera networks is also possible. A novel time difference estimation algorithm is developed that involves estimating time difference during tracking. Individual particle filters compute time difference values. These noisy values are inputted to time difference estimator, which smoothes the computed time differences and feeds the smoothed value to the particle filters. The particle filter compensates this time difference, while tracking targets. Gaussian mixture kernel density estimator is demonstrated to perform better than Kalman filter and cummulative running average estimators. This estimator is used as the time difference estimator in the experiments. When the magnitude of time difference between frames exceeds 1 second, the performance of the tracking algorithm decrease to unacceptable levels such that reliable time difference computations cannot be generated by the trackers.

Finally, velocity and position states of the particle filter trackers are divided as linear and nonlinear states in a Rao-Blackwellized fashion. Velocity state of particles is marginalized by a Kalman filter, while their position state is filtered by the particle filter. The performance of the Rao-Blackwellized algorithm is slightly greater than that of the original algorithm regardless of the number of particles. The performance difference between both algorithms increased from 1% to 2%, when number of particles are decreased from 2000 and 1000 to 500 and 250. The increase in performance is achieved at the cost of an increase in processing time as a Kalman filter is run for each particle in the Rao-Blackwellized algorithm. Additionally, the localization accuracies of both algorithms are demonstrated to be almost the same.

REFERENCES

[1] M. Valera and S.A. Velastin, "Intelligent distributed surveillance systems: a review", *IEE Proc. Vis. Image Signal Process.*, vol. 152, no. 2, pp. 192-204, April, 2005.

[2] A.R. Dick and M.J. Brooks, "Issues in automated visual surveillance", *Proc. 7th Biennial Australian Patt. Recog. Society Conf.*, pp. 195-203, 2003.

[3] N. Haering, P.L. Venetianer and A. Lipton, "The evolution of video surveillance: an overview", *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 279-290, August, 2008.

[4] Video tracking, http://en.wikipedia.org/wiki/Video_tracking, last accessed in 18 August 2012.

[5] G. R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface", *Intel Technology Journal*, 1998.

[6] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", *Proceedings of the 1981 DARPA Image Understanding Workshop*, pp. 121-130, 1981.

[7] T. P. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranov and V. Eruhimov, "Computer Vision Workload Analysis: Case Study of Video Surveillance Systems", *Intel Technology Journal*, vol. 9, no. 2, 2005.

[8] Y. Wu, T.Yu and G. Hua, "Tracking Appearances with Occlusions", *CVPR*, pp. 789-795, 2003.

[9] J. Shi and C. Tomasi, "Good Features to Track", *Computer Vision and Pattern Recognition*, pp. 593-600, 1994.

[10] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, vol. 60, no.2, pp. 91-110, 2004.

[11] H. Bay, T. Tuytelaars and L. Van Gool, "SURF: Speeded Up Robust Features", *European Conference on Computer Vision*, pp. 404-417, 2006.

[12] D. Comaniciu, V. Ramesh and P. Meer, "Real-Time Tracking of Non-Rigid Objects Using Mean Shift", *CVPR*, vol. II, pp. 142-149, June, 2000.

[13] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multicamera people tracking with a probabilistic occupancy map," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 267-282, February 2008.

[14] S. M. Khan and M. Shah, "Tracking multiple occluding people by localizing on multiple scene planes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 133-146, March 2009.

[15] R. Eshel and Y. Moses, "Tracking in a dense crowd using multiple cameras," *International Journal on Computer Vision*, vol. 88, pp. 129-143, 2010.

[16] W. Du and J. Piater, "Multi-camera people tracking by collaborative particle filters and principal axis-based integration," in *ACCV*, Y. Yagi, S. B. Kang, I. S. Kweon, and H. Zha, Eds., vol. 4843, pp. 365-374. Springer, Heidelberg, 2007.

[17] E. Maggio and A. Cavallaro, *Video tracking: theory and practice*, Wiley.com, 2011.

[18] Z. Yue, S. K. Zhou, and R. Chellappa, "Robust two camera tracking using homography," in *ICASSP*, 2004.

[19] K. Kim and L. S. Davis, "Multi-camera Tracking and Segmentation of Occluded People on Ground Plane Using Search-Guided Particle Filtering", *Nineth European Conference on Computer Vision*, Graz, Austria, pp. 98-109, 2006.

[20] A. Azarbayejani and A. Pentland, "Real-Time Self-Calibrating Stereo Person Tracking Using 3D Shape Estimation from Blob Features", 1996, Technical Report MIT Media Laboratory.

[21] J. Black and T. Ellis, "Multi camera image tracking," *Image and Vision Computing*, vol. 24, pp. 1256-1267, 2006.

[22] C. R. del Blanco, R. Mohedano, N. Garcia, L. Salgado, and F. Jaureguizar, "Color-based 3d particle filtering for robust tracking in heterogeneous environments," in *Second ACM/IEEE International Conference on Distributed Smart Cameras*, 2008, pp. 1-10.

[23] A. Tyagi, M. Keck, J. W. Davis and G. Potamianos, "Kernel-Based 3D Tracking", *CVPR*, pp. 1-8, 2007.

[24] M. Isard and A. Blake, "CONDENSATION - conditional density propagation for visual tracking", *International Journal of Computer Vision*,vol. 29, no. 1, pp. 5-28, 1998.

[25] A. Mittal and L. S. Davis, "M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene", *International Journal of Computer Vision*, vol. 51, no. 3, pp. 189-203, February, 2003.

[26] S. M. Khan and M. Shah, "A Multiview Approach to Tracking People in Crowded Scenes Using a Planar Homography Constraint", *ECCV*, pp. 133-146, 2006.

[27] G. Welch and G. Bishop, "An Introduction to the Kalman Filter", Department of Computer Science, University of North Carolina, 1995.

[28] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics*, MIT Press, Cambridge, MA, 2005.

[29] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Pearson Education Inc., NJ, 2003.

[30] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/NonGaussian Bayesian Tracking", *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 174-188, February, 2002.

[31] D. Beymer and J. Malik, "Tracking vehicles in congested traffic", *Proceedings of IEEE Intelligent Vehicles Symposium*, pp. 130-135, 1996.

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

[33] A. M. Baumberg, *Learning Deformable Models for Tracking Human Motion*, PhD Thesis, School of Computer Studies, University of Leeds, October 1995.

[34] H. Wang, D. Suter, K. Schindler and C. Shen, "Adaptive object tracking based on an effective appearance filter", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1661-1667, September, 2007.

[35] A. D. Jepson, D. J. Fleet and T. F. El-Maraghi, "Robust Online Appearance Models for Visual Tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, October, 2003.

[36] B. Han, D. Comaniciu, Y. Zhu and L. S. Davis, "Sequential Kernel Density Approximation and Its Application to Real-Time Visual Tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 7, July, 2008.

[37] S. J. McKenna, Y. Raja and S. Gong, "Tracking color objects using adaptive mixture models", *Image and Vision Computing*, vol. 17, pp. 225-231, 1999.

[38] P. Perez, C. Hue, J. Vermaak and M. Gangnet, "Color-Based Probabilistic Tracking", *ECCV*, pp. 661-675, 2002.

[39] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti and R. Bolle, "Appearance Models for Occlusion Handling", *In Second International Workshop on Performace Evaluation of Tracking and Surveillance Systems*, 2001.

[40] P. F. Gabriel, J. G. Verly, J. H. Piater, and A. Genon, "The state of the art in multiple object tracking under occlusion in video sequences", *in Proceedings of ACIVS*, September, 2003.

[41] D. Meyer, J. Denzler, and H. Niemann, "Model based extraction of articulated objects in image sequences for gait analysis", *ICIP*, vol. 2, pp. 78-81, 1998.

[42] C. Stauffer and W.E.L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking", *CVPR*, Fort Collins, CO, USA, 1999.

[43] D.S. Lee, "Effective Gaussian Mixture Learning for Video Background Subtraction", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 827-832, May, 2005.

[44] T. H. Chao, B. Lau, and Y. Park, "Vehicle detection and classification in shadowy traffic images using wavelets and neural networks", *Proc. SPIE*, vol. 2902, pp. 136-147, 1997.

[45] G. Zhang, R. P. Avery, and Y. Wang, "A video-based vehicle detection and classification system for real-time traffic data collection using uncalibrated video cameras", Transportation Research Board, vol. 1993, pp. 138-147, 2007.

[46] A. H. S. Lai, and N. H. C. Yung, "A fast and accurate scoreboard algorithm for estimating stationary backgrounds in an image sequence", *Proc. IEEE Intl. Symp. Circuits and Systems*, Monterey CA 1998, vol. 4, pp. 241-244.

[47] S. N. Lim, A. Mittal, L. S. Davis, and N. Paragios, "Fast illumination-invariant background subtraction using two views: error analysis, sensor placement and applications", *CVPR*, San Diego CA 2005, vol. 1, pp. 1071-1078.

[48] Y. A. Ivanov, A. F. Bobick, and J. Liu, "Fast lighting independent background subtraction", *International Journal of Computer Vision*, vol. 37, no. 2, pp. 199-207, June, 2000.

[49] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale and S. Shafer, "Multi-camera multi-person tracking for EasyLiving", *3rd IEEE Intl. Workshop Visual Surveillance*, Dublin, Ireland, 2000, pp. 3-10.

[50] O. Javed, K. Shafique and M. Shah, "A hierarchical approach to robust background subtraction using color and gradient information", *Proc. Workshop Motion and Video Computing*, Los Alamitos, CA, 2002, pp. 22-27.

[51] A. O. Ercan, "Object Tracking via a Collaborative Camera Network", Ph.D. Thesis, Stanford University, 2007.

[52] W. Hu, X. Zhou, M. Hu and S. Maybank, "Occlusion Reasoning for Tracking Multiple People", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 1. January, 2009.

[53] S. Kwak, W. Nam, B. Han, and J. H. Han, "Learning Occlusion with Likelihoods for Visual Tracking", *ICCV*, pp. 1551-1558, 2011.

[54] A. O. Ercan, D. B.-R. Yang, A. E. Gamal, and L. J. Guibas, "Optimal placement and selection of camera network nodes for target localization," in *Proceedings of DCOSS*, pp. 389-404, June 2006.

[55] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House, 1999.

[56] I. Saleemi, L. hartung and M. Shah, "Scene understanding by statistical modeling of motion patterns", in *CVPR*, 2010, pp. 2069-2076.

[57] N. Anjum and A. Cavallaro, "Multifeature Object Trajectory Clustering for Video Analysis", *IEEE Transactions on Circuits amd Systems for Video Technology*, vol. 18, no. 11, pp. 1555-1564, 2008.

[58] E. Maggio and A. Cavallaro, "Learning Scene Context for Multiple Object Tracking", *IEEE Transactions on Image Processing*, vol. 18, no. 8, pp. 1873-1884, 2009.

[59] Microsoft Corporation, "Kinect sensor," 2012, http://msdn.microsoft.com/en-us/library/hh438998, last accessed in 8 October 2013.

[60] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE Multimedia*, pp. 4-10, February 2012.

[61] D. M. Gavrila and L. S. Davis, "3-d model-based tracking of humans in action: a multi-view approach," in *CVPR*, 1996, pp. 73-80.

[62] A. A. Alatan, Y. Yemez, U. Gudukbay, X. Zabulis, K. Muller, C. E. Erdem, C. Weigel, and A. Smolic, "Scene representation technologies for 3DTV - A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1587-1605, 2007.

[63] D. Focken and R. Stiefelhagen, "Towards vision-based 3d people tracking in a smart room," in *Proc. Fourth Intl. Conf. on Multimodal Interfaces*, 2002.

[64] R. Vos and W. Brink, "Multi-view 3d position estimation of sports players," in *Proceedings of the Twenty-First Annual Symposium of the Pattern Recognition Association of South Africa*, 2010.

[65] T. Zhao, R. Nevatia, and B. Wu, "Segmentation and tracking of multiple humans in crowded environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 7, 2008.

[66] C. Canton-Ferrer, J. R. Casas, M. Pardas, and R. Sblendido, "Particle filtering and sparse sampling for multi-person 3d tracking," in *ICASSP*, 2007, pp. I-913-I-916.

[67] J. Ferryman, "Pets 2006," March 2009, http://www.cvg.rdg.ac.uk/PETS2006/, last accessed in 8 October 2013.

[68] J. Ferryman, "Pets 2009," March 2009, http://www.cvg.rdg.ac.uk/PETS2009/, last accessed in 8 October 2013.

[69] Z. Kalal, K. Mikolajczyk, and J. Matas, "P-n learning: Bootstrapping binary classifiers by structural constraints," *CVPR*, 2010.

[70] K. Nummiaro, Esther Koller-Meier and Luc Van Gool, "An Adaptive Color-Based Particle Filter", *Image and Vision Computing*, vol. 21, no. 1, pp. 99-110, January, 2003.

[71] J. Yao and J.-M. Odobez, "Multi-person bayesian tracking with multiple cameras," *Multi-Camera Networks: Principles and Applications*, pp. 363-388, 2009.

[72] G. D. Sullivan, K. D. Baker, A. D. Worrall, C. I. Attwood, and P. M. Remagnino, "Model-based vehicle detection and classification using orthographic approximations," *Image and Vision Computing*, vol. 15, no. 8, pp. 649-654, 1997.

[73] Alper Yilmaz, Omar Javed, and Mubarak Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006.

[74] Y. Caspi and M. Irani, "Spatio-Temporal Alignment of Sequences", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1409-1424, 2002.

[75] V. I. Morariu and O. I. Camps, "Modeling correspondences for multi-camera tracking using nonlinear manifold learning and target dynamics," in *CVPR*, 2006.

[76] R. Hartley and A. Zisserman, *Multi View Geometry in Computer Vision*, Cambridge University Press, Second Edition, 2004.

[77] A. Papoulis and S. U. Pillai, Probability, *Random Variables and Stochastic Processes*, McGraw-Hill Higher Education, fourth edition, 2002.

[78] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *ICCV*, 2009, pp. 261-268.

[79] L. Leal-Taixe, G. Pons-Moll, and B. Rosenhahn, "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker," in *ICCV Workshops*, 2011, pp. 120-127.

[80] Y. Bar-Shalom, P. K. Willet, and X. Tian, *Tracking and Data Fusion: A Handbook of Algorithms*, YBS Publishing, 2011.

[81] A. O. Ercan, A. El Gamal, and L. J. Guibas, "Object tracking in the presence of occlusions using multiple cameras: A sensor network approach," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 2, p. 16, 2013.

[82] T. Kailath, "The divergence and bhattacharyya distance measures in signal selection," *IEEE Transactions on Communication Technology*, vol. 15, no. 1, pp. 52-60, February 1967.

[83] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, John Wiley and Sons, 2001.

[84] T. Bengtsson, P. Bickel, and B. Li, "Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems," *Probability and statistics: Essays in honor of David A. Freedman*, vol. 2, pp. 316-334, 2008.

[85] I. W. McKeague and W. Wefelmeyer, "Markov chain monte carlo and rao-blackwellization," *Journal of statistical planning and inference*, vol. 85, no. 1, pp. 171-182, 2000.

[86] Y. Kosuge, "Non-process-noise tracking filter using a constant velocity model," in *SICE Annual Conference*, August 2008, pp. 2670-2674.

[87] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 319-336, February 2009.

[88] J. Berclaz, "CVLAB", http://cvlab.epfl.ch/data/pom/, 20 April 2012, last accessed in 8 October 2013.

[89] F. Sivrikaya and B. Yener, "Time Synchronization in Sensor Networks: A Survey", *IEEE Network*, vol. 18, no. 4, pp. 45-50, 2004.

[90] G. P. Stein, "Tracking from Multiple View Points: Self-calibration of Space and Time", *Proceedings of DARPA IU Workshop*, pp. 1037-1042, 1998.

[91] Y. Caspi, D. Simakov and M. Irani, "Feature-Based Sequence-to-Sequence Matching", *International Journal of Computer Vision*, vol. 68, no. 1, 2006.

[92] A. Whitehead, R. Laganiere and P. Bose, "Temporal Synchronization of Video Sequences in Theory and in Practice", *Proceedings of the IEEE Workshop on Motion and Video Computing*, 2005.

[93] Thomas Schon, Fredrik Gustafsson, and P-J. Nordlund. "Marginalized particle filters for mixed linear/nonlinear state-space models." *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2279-2289, 2005.

[94] Zhe Chen, "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond", *Statistics*, vol. 182, no. 1, pp. 1-69, 2003.

[95] J. Kim and I.-S. Kweon, "Rao-Blackwellized particle filter for Gaussian mixture models and application to visual tracking", in *ICASSP*, pp. 1173-1176, May 2011.

[96] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.

APPENDIX A

SEQUENTIAL BAYESIAN FILTERING

The state at time step t is denoted as x_t such that $t = 0, 1, ..., N$. The state sequence $\{x_0, x_1, ..., x_N\}$ is written shortly as $\{x_{0:N}\}$, and similarly the observation sequence is written as $\{z_{1:N}\}$.These sequences correspond to the joint event of occurrences of states and observations in the given order. The state evolution in time is represented by

$$x_t = f_t(x_t, w_t) \tag{A.1}$$

where $\mathbf{f_t}$ is a *nonlinear* function of state evolution and $\{w_t\}_{t=1}^{N}$ is the state evolution noise sequence that is assumed to be i.i.d and white. The observation is generated through the following relation

$$z_t = h_t(x_t) + v_t \tag{A.2}$$

where h_t is a nonlinear function of state evolution and $\{v_t\}_{t=1}^{N}$ is the observation noise sequence that is assumed to be i.i.d and white. The state evolution noise and observation noise are assumed to be independent. The purpose is to make inferences from the posterior probability density function (pdf), $p(x_{0:t}|z_{1:t})$ at each time step t sequentially beginning from $p(x_0)$. The initial pdf is assumed to be known a priori that is either taken as uniform or other distribution close to uniform to account for the a priori information about target state. Using the Bayes' rule, the posterior pdf is written as

$$p(x_{0:t}|z_{1:t}) = \frac{p(z_{1:t}|x_{0:t})p(x_{0:t})}{p(z_{1:t})}. \tag{A.3}$$

The first pdf in the numerator at the right side of equation (A.3) is simplified as

$$\begin{aligned} p(z_{1:t}|x_{0:t}) &= p(z_k|z_{1:t-1}, x_{0:t})p(z_{1:t-1}|x_{0:t}) && \text{(A.4)} \\ &= p(z_t|x_t)p(z_{1:t-1}|x_{0:t}) && \text{(A.5)} \\ &= p(z_t|x_t)p(z_{1:t-1}|x_{0:t-1}). && \text{(A.6)} \end{aligned}$$

Equation (A.5) is obtained from Equation (A.4), since observation at t depends only on the state at t according to (A.2) where the noise sequence is white. Equation (A.6) follows, since x_t provides no additional information. If this decomposition is continued [96],

$$p(z_{1:t}|x_{0:t}) = \prod_{t=1}^{N} p(z_t|x_t) \tag{A.7}$$

is obtained. Likewise, the second pdf in the numerator at the right side of equation (A.3) is simplified as [96]

$$p(x_{0:t}) = p(x_k|x_{0:t-1})p(x_{0:t-1}) \tag{A.8}$$

$$= p(x_t|x_{t-1})p(x_{0:t-1}) \tag{A.9}$$

$$= p(x_0)\prod_{t=1}^{N} p(x_t|x_{t-1}). \tag{A.10}$$

Equation (A.9) followed equation (A.8), due to Markovian property of the state sequence. More specifically, if the state is complete [28], then no state prior to x_{k-1} can provide additional information about current state once x_{t-1} is known. In other words, x_{t-1} is a sufficient statistic of all previous states [28]. Combining equations (A.7) and (A.10), the posterior pdf is expressed as,

$$p(x_{0:t}|z_{1:t}) = Cp(x_0)\prod_{t=1}^{N} p(z_t|x_t)p(x_t|x_{t-1}) \tag{A.11}$$

where C is a constant equal to multiplicative inverse of $p(z_{1:t})$. The posterior pdf can be computed sequentially according to,

$$p(x_{0:t}|z_{1:t}) = \frac{p(z_t|x_t)p(x_t|x_{t-1})}{p(z_t|z_{1:t-1})}p(x_{0:t-1}|z_{1:t-1}). \tag{A.12}$$

The posterior pdf is constructed sequentially as in equation (A.12) and the state sequence that minimizes a cost function is computed at each time step.

APPENDIX B

THE BAYES FILTER

The Bayes filter is included here for the sake of completeness. Different from sequential Bayesian filtering algorithm, Bayes filter aims to construct the posterior pdf, $p(x_t|z_{1:t})$. The derivation is given in the following lines.

$$p(x_t|z_{1:t}) = \frac{p(x_t, z_{1:t})}{p(z_{1:t})} \tag{B.1}$$

$$= \frac{p(z_t, z_{1:t-1}|x_t)p(x_t)}{p(z_t, z_{1:t-1})} \tag{B.2}$$

$$= \frac{p(z_t|z_{1:t-1}, x_t)p(z_{1:t-1}|x_t)p(x_t)}{p(z_t|z_{1:t-1})p(z_{1:t-1})} \tag{B.3}$$

Replacing $p(z_{1:t-1}|x_t) = \frac{p(x_t|z_{1:t-1})p(z_{1:t-1})}{p(x_t)}$ in equation (B.3), the derivation continues as,

$$p(x_t|z_{1:t}) = \frac{p(z_t|z_{1:t-1}, x_t)p(x_t|z_{1:t-1})p(z_{1:t-1})p(x_t)}{p(z_t|z_{1:t-1})p(z_{1:t-1})p(x_t)} \tag{B.4}$$

$$= \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{B.5}$$

where equation (B.5) is obtained after cancelations and the assumption that the observation noise sequence is white. More specifically, the observations are assumed to be correlated, since the states are correlated [94],[28]. Given states, the observations become uncorrelated as observation noise sequence is assumed to be white (A.2). The prior pdf is expressed as,

$$p(x_t|z_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1}. \tag{B.6}$$

The Bayes filter algorithm has 2 steps: prediction and correction. In the prediction step the prior pdf is generated solving the integral equation (B.6). The posterior pdf is generated from equation (B.5) by incorporating the likelihood pdf, $p(z_t|x_t)$, to the prior pdf in the correction step. More information can be found on Bayes filter in [28],[94],[30].

APPENDIX C

FOUNDATIONS OF PARTICLE FILTERING

C.1 Monte Carlo Sampling Approximation

Monte Carlo sampling approximation is about taking a number of random samples from the domain of a pdf such that samples become denser near high probability density regions. A pdf, say $p(x)$, is represented by these samples as,

$$p(x) \approx \frac{1}{N_0} \sum_{i=1}^{N_0} \delta(x - x^{(i)}) \tag{C.1}$$

where N_0 is the number of samples. An integral equation, such as $\int_X f(x)p(x)dx$, is approximated by replacing equation (C.1) as,

$$\int_X f(x)p(x)dx \approx \frac{1}{N_0} \sum_{i=1}^{N_0} f(x^{(i)}) \tag{C.2}$$

where X is the domain of the pdf.

C.2 Importance Sampling

In some cases, the pdf cannot be sampled directly. The strategy to overcome this situation is to sample another pdf and represent the original pdf by a set of weighted samples. This another pdf is called proposal distribution.

$$\int_X f(x)p(x)dx = \int_X \frac{p(x)}{q(x)} f(x)q(x)dx \tag{C.3}$$

$$= \int_X w(x)f(x)q(x)dx \tag{C.4}$$

$$\approx \frac{1}{N_0} \sum_{i=1}^{N_0} w(\hat{x}^{(i)})f(\hat{x}^{(i)}) \tag{C.5}$$

such that $\hat{x}^{(i)}$ is sampled from the proposal distribution, $q(x)$, $\hat{x}^{(i)} \sim q(x)$, and $w(x)$ is the weight. The support of the proposal distribution must cover the support of the original pdf [94]. In practice, both pdfs to have close values is desirable. Those cases where one density having very small values and the other having larger values results in over-emphasizing or under-emphasizing random samples. Therefore, choosing better proposal distribution results in greater accuracy.

C.3 Derivation of Particle Filter Algorithm

The particle filter algorithm represents the posterior pdf $p(x_{0:t}|z_{1:t})$ using a set of random samples. Instead of direct Monte Carlo sampling from this distribution, samples are going to be drawn from a proposal distribution, $q(x_{0:t}|z_{1:t})$. Assume that the proposal distribution can be factorized as,

$$q(x_{0:t}|z_{1:t}) = q(x_t|x_{0:t-1}, z_{1:t})q(x_{0:t-1}|z_{1:t-1}). \tag{C.6}$$

In other words, z_t assumed to provide no additional information to the state $x_{0:t-1}$ as suggested by $q(x_{0:t-1}|z_{1:t}) = q(x_{0:t-1}|z_{1:t-1})$. The plan is to represent the posterior pdf with a set of random samples drawn from $q(x_{0:t}|z_{1:t})$ that are weighted by their ratio as in equation (C.4). The weight at t is defined as,

$$w_t \propto \frac{p(x_{0:t}|z_{1:t})}{q(x_{0:t}|z_{1:t})} \tag{C.7}$$

$$= \frac{p(z_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|z_{1:t-1})}{q(x_t|x_{0:t-1}, z_{1:t})q(x_{0:t-1}|z_{1:t-1})} \tag{C.8}$$

$$= w_{t-1}\frac{p(z_t|x_t)p(x_t|x_{t-1})}{q(x_t|x_{0:t-1}, z_{1:t})}. \tag{C.9}$$

Equation (C.8) followed equation (C.7) by replacing the equation (A.12) and (C.6). Choosing the state transition pdf as the proposal density, $q(x_t|x_{0:t-1}, z_{1:t}) = p(x_t|x_{t-1})$, the weight becomes proportional to

$$w_t \propto w_{t-1}p(z_t|x_t). \tag{C.10}$$

Therefore, the posterior density can be expressed as,

$$p(x_{0:t}|z_{1:t}) \propto w_{t-1}p(z_t|x_t)p(x_t|x_{t-1})q(x_{0:t-1}|z_{1:t-1}) \tag{C.11}$$

Assume that a set of samples with associated weights are available, $\{x^{(i)}_{0:t-1}, w^{(i)}_{t-1}\}_{i=0}^{N_p}$, that represents the posterior probability at $t-1$ as,

$$p(x_{0:t-1}|z_{1:t-1}) \approx \sum_{i=1}^{N_p} w^{(i)}_{t-1}\delta(x_{0:t-1} - x^{(i)}_{0:t-1}) \tag{C.12}$$

where N_p is the number of particles, $x^{(i)}_{0:t-1}$ is drawn from $q(x_{0:t-1}|z_{1:t-1})$ and $w^{(i)}_{t-1}$ is computed as in equation (C.10). Replacing equation (C.12) into equation (C.11),

$$p(x_{0:t}|z_{1:t}) \approx \sum_{i=1}^{N_p} w^{(i)}_{t-1}p(z_t|x_t)p(x_t|x_{t-1})\delta(x_{0:t-1} - x^{(i)}_{0:t-1}) \tag{C.13}$$

$$= \sum_{i=1}^{N_p} w^{(i)}_{t-1}p(z_t|x_t)p(x_t|x^{(i)}_{t-1})\delta(x_{0:t-1} - x^{(i)}_{0:t-1}) \tag{C.14}$$

The state transition pdf (at the same time the proposal pdf), $p(x_t|x^{(i)}_{t-1})$, is sampled by sampling the state evolution noise in

$$p(x_t|x^{(i)}_{t-1}) = \int p(x_t|x^{(i)}_{t-1}, u_t)p(u_t|x^{(i)}_{t-1})du_t \tag{C.15}$$

$$= \int p(x_t|x^{(i)}_{t-1}, u_t)p(u_t)du_t \tag{C.16}$$

$$= \int \delta(x_t - f_t(x^{(i)}_{t-1}, u_t))p(u_t)du_t. \tag{C.17}$$

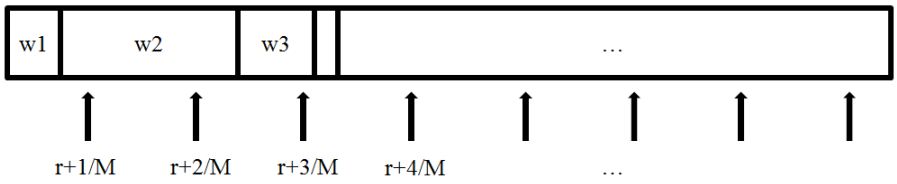


Figure C.1: Illustration of variance reduction technique used in this thesis work.

Thus, the state at t, x_t can be sampled according to the relation

$$x_t^{(i)} = f_t(x_{t-1}^{(i)}, u_t^{(i)}) \tag{C.18}$$

as Monte Carlo samples, $\{x_{0:t}^{(i)}\}_{i=1}^{N_p}$, are generated using samples from u_t. Replacing the samples generated by equation (C.18) into (C.14),

$$p(x_{0:t}|z_{1:t}) \approx \sum_{i=1}^{N_p} w_{t-1}^{(i)} p(z_t|x_t^{(i)}) \delta(x_{0:t} - x_{0:t}^{(i)}) \tag{C.19}$$

$$= \sum_{i=1}^{N_p} w_t^{(i)} \delta(x_{0:t} - x_{0:t}^{(i)}) \tag{C.20}$$

where

$$w_t^{(i)} = w_{t-1}^{(i)} p(z_t|x_t^{(i)}) \tag{C.21}$$

is used. This is the derivation of the standard particle filter. The algorithm pseudo-code is available in [29], [28], [30] and [94].

C.4 Variance Reduction

Particle filters exhibit weight degeneracy as time evolves. Weight degeneracy is the case when only a few particles have non-zero weights. To overcome this problem, variance reduction techniques are developed by previous researchers. More information on this topic can be found in [30], [94] and [28]. The variance reduction technique used in this thesis is best presented in [28] for the interested reader.

One of the popular variance reduction technique, which is used in this thesis work, is illustrated in Figure C.1. In this figure, particle weights, denoted by w, are proportional to the corresponding segment widths. A random number, r, is drawn from uniform distribution between the interval $[0, 1/M]$ where M is the number of particles. This random number eliminates the first particle, if its weight is small enough. The first particle is eliminated in Figure C.1, since r is larger than its weight, $w1$. Arrows are placed $1/M$ apart from each other as shown in Figure C.1. Those particles that are indicated by the arrows are kept and those that are not are eliminated. If arrows visit a segment more than once as it does the second particle, than this particle is reproduced the number of times arrows visit them.

VITA

PERSONAL INFORMATION

Surname, Name: Topçu, Osman
Nationality: Turkish (TC)
Date and Place of Birth: 20 May 1980, Fethiye/Muğla
Marital Status: Single
Mobile Phone: +90 535 229 4114
Email: osman.topcu@tubitak.gov.tr

EDUCATION

Degree	Institution	Year of Graduation
MS	Bilkent Univ. Electrical and Electronics Engineering	2006
BS	Bilkent Univ. Electrical and Electronics Engineering	2003

WORK EXPERIENCE

Year	Place	Enrollment
2010-Present	TUBITAK UZAY	Senior Researcher
2008-2009	METU EEE	Software Developer
2006	Aselsan	Engineer

FOREIGN LANGUAGES

Advanced English, Basic German

PUBLICATIONS

1. Topçu O., Ercan A. Ö., Alatan A., "Görünüş Temelli Örtüşme Gözeten 3B Nesne Takibi", 20. SIU Konferansı, 2012.
2. Topçu O., Kalem A., Esen E. "Hareketli Kamerada Arka Plan Çıkarma", 21. SIU Konferansı, 2013.

HOBBIES

Basketball, Music, Movies